Günter Rudolph Thomas Jansen
Simon Lucas Carlo Poloni
Nicola Beume (Eds.)

LNCS 5199

Parallel Problem Solving from Nature – PPSN X

10th International Conference
Dortmund, Germany, September 2008
Proceedings

Springer

Günter Rudolph Thomas Jansen
Simon Lucas Carlo Poloni Nicola Beume (Eds.)

Parallel Problem Solving from Nature – PPSN X

10th International Conference
Dortmund, Germany, September 13-17, 2008
Proceedings

Springer

Volume Editors

Günter Rudolph
Thomas Jansen
Nicola Beume
Technische Universität Dortmund
Fakultät für Informatik
44221 Dortmund, Germany
E-mail: {guenter.rudolph, thomas.jansen, nicola.beume}@tu-dortmund.de

Simon Lucas
University of Essex
Department of Computing and Electronic Systems
Colchester, Essex CO4 3SQ, UK
E-mail: sml@essex.ac.uk

Carlo Poloni
Università degli Studi di Trieste
Dipartimento di Ingegneria Meccanica
34127 Trieste, Italy
E-mail: poloni@univ.trieste.it

Library of Congress Control Number: 2008934679

CR Subject Classification (1998): F.1-2, C.1.4, D.1-3, I.2.8, I.2.6, I.2.11, J.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN	0302-9743
ISBN-10	3-540-87699-5 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-87699-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12517503 06/3180 5 4 3 2 1 0

Preface

The first major gathering of people interested in discussing natural paradigms and their application to solve real-world problems in Europe took place at Dortmund, Germany, in 1990. What was planned originally as a small workshop with about 30 participants finally grew into an international conference named Parallel Problem Solving from Nature (PPSN) with more than 100 participants. The interest in the topics of the conference has increased steadily ever since leading to the pleasant necessity of organizing PPSN conferences biennially within the European region. After visiting Brussels (1992), Jerusalem (1994), Berlin (1996), Amsterdam (1998), Paris (2000), Granada (2002), Birmingham (2004), and Reykjavik (2006), PPSN returned to its birthplace in Dortmund to celebrate its 10th anniversary in 2008.

Without any doubt the PPSN conference series evolved to be one of the most respected and highly regarded conferences on natural computing. Therefore we are very pleased to present the proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN X) to the scientific community. This year we received 206 submissions with authors from 26 countries spread over Africa, America, Asia, Australia and Europe. From these submissions the Program Chairs selected the top 114 papers after an extensive peer-review process. Not all decisions were easy to make but in all cases we benefited greatly from the careful reviews provided by the international Program Committee. We requested four reviews for each submission leading to a total of 824 requests for reviews. Thanks to these reviews we were able to decide about acceptance on a solid basis.

The papers included in these proceedings have been assigned to six fuzzy clusters (formal theory, new techniques, experimental analysis, multiobjective optimization, hybrid methods, and applications) that can hardly reflect the true variety of research topics presented in the proceedings at hand. Following the tradition and spirit of PPSN, all papers were presented as posters. The 8 poster sessions consisting of about 14 papers each were compiled orthogonally to the fuzzy clusters mentioned above to cover the range of topics as widely as possible. As a consequence, participants with different interests would find some relevant papers in every session and poster presenters were able to discuss related work in sessions different to their own. As usual, the conference also included one day with workshops (Saturday), one day with tutorials (Sunday), and three invited plenary talks (Monday to Wednesday) for free.

Needless to say, the success of such a conference depends on authors, reviewers, and organizers. We are grateful to all authors for submitting their best and latest work, to all the reviewers for the generous way they spent their time and provided their valuable expertise in preparing these reviews, to the workshop organizers and tutorial presenters for their valorizing contributions to the conference event, and to the local organizers who helped to make PPSN X happen.

Last but not least, we would like to thank the Collaborative Research Center ‘Computational Intelligence’ (SFB 531) at the Dortmund University of Technology and the Deutsche Forschungsgemeinschaft (DFG) for financial support as well as the Gesellschaft für Informatik (GI) for administrative support in international monetary transactions.

July 2008

Günter Rudolph
Thomas Jansen
Simon Lucas
Carlo Poloni
Nicola Beume

Organization

PPSN X was organized and hosted by the Computational Intelligence Group of the Lehrstuhl für Algorithm Engineering, Fakultät für Informatik, Technische Universität Dortmund (TU Dortmund), Germany. The conference took place in the Kongresszentrum Westfalenhallen, Dortmund.

Conference Committee

General Chair	Günter Rudolph, TU Dortmund, Germany
Program Chairs	Thomas Jansen, TU Dortmund, Germany Simon Lucas, University of Essex, UK Carlo Poloni, Università degli Studi di Trieste, Italy
Workshop Chair	Mike Preuss, TU Dortmund, Germany
Tutorial Chair	Boris Naujoks, TU Dortmund, Germany
E-Affairs Chair	Nicola Beume, TU Dortmund, Germany
Local Organization	Gundel Jankord, TU Dortmund, Germany Oliver Kramer, TU Dortmund, Germany
Honorary Chair	Hans-Paul Schwefel, TU Dortmund, Germany

Steering Committee

David W. Corne	Heriot-Watt University Edinburgh, UK
Kenneth De Jong	George Mason University Fairfax, USA
Agoston E. Eiben	Vrije Universiteit Amsterdam, The Netherlands
Juan J. Merelo Guervós	Universidad de Granada, Spain
Günter Rudolph	TU Dortmund, Germany
Thomas P. Runarsson	Háskóli Íslands Reykjavík, Iceland
Marc Schoenauer	INRIA Saclay – Île-de-France, France
Xin Yao	University of Birmingham, UK

Workshops

Theory of Randomized Search Heuristics 2008
Frank Neumann and Benjamin Doerr

Hyper-heuristics Automating the Heuristic Design Process
Gabriela Ochoa and Ender Özcan

Set-Based Evolution
Gideon Avigad

Computational Intelligence and Games
Simon Lucas and Thomas P. Runarsson

Tutorials

A Unified Approach to Evolutionary Computation
Kenneth De Jong

Swarm Intelligence: Particle Swarm Optimization and Ant Colony Optimization
Christian Blum

Multiobjective Evolutionary Algorithms
Eckart Zitzler

Transportation and Logistics
Jens Gottlieb

Bioinformatics
David W. Corne

Games
Simon Lucas

Experimental Research in Natural Computation
Thomas Bartz-Beielstein and Mike Preuss

Computational Complexity of Evolutionary Computation in Combinatorial Optimization
Frank Neumann and Carsten Witt

Evolution Strategies and Related Estimation of Distribution Algorithms
Nikolaus Hansen and Anne Auger

Program Committee

Uwe Aickelin
Enrique Alba Torres
Dirk V. Arnold
Daniel Ashlock
Anne Auger
Wolfgang Banzhaf
Thomas Bartz-Beielstein
Peter J. Bentley
Nicola Beume
Hans-Georg Beyer
Mark Bishop
Christian Blum
Yossi Borenstein
Peter A.N. Bosman
Jürgen Branke
Dimo Brockhoff
Bobby D. Bryant
Larry Bull
John A. Bullinaria
Edmund K. Burke
Erick Cantú-Paz
Steve Cayzer
Uday K. Chakraborty
Sung-Bae Cho
Carlos A. Coello Coello
Pierre Collet
David W. Corne
Carlos Cotta
Peter I. Cowling
Kenneth De Jong
Kalyanmoy Deb
Benjamin Doerr
Marco Dorigo
Stefan Droste
Agoston E. Eiben
Michael T.M. Emmerich
Andries P. Engelbrecht
Thomas M. English
Anna I. Esparcia Alcázar
Udo Feldkamp
Gary B. Fogel
Carlos M. Fonseca
Olivier François
Tobias Friedrich
Marcus Gallagher
Jonathan M. Garibaldi
Mario Giacobini
Kyriakos Giannakoglou
Jens Gottlieb
Garrison W. Greenwood
Roderich Groß
Steven Gustafson
Hisashi Handa
Julia Handl
Nikolaus Hansen
Emma Hart
Michael Herdy
Philip F. Hingston
Frank Hoffmann
Jeffrey Horn
Evan J. Hughes
Eyke Hüllermeier
Christian Igel
Pedro Isasi
Jens Jägersküpper
Wilfried Jakob
Márk Jelasity
Yaochu Jin
Bob John
Bryant A. Julstrom
Andy J. Keane
Maarten A. Keijzer
Robert E. Keller
Graham Kendall
Joshua D. Knowles
Wolfgang Konen
Oliver Kramer
Natalio Krasnogor
Andreas Kroll
Saku Kukkonen
William B. Langdon
Pier Luca Lanzi
Pedro Larrañaga
Ulrich Lehmann
Philipp Limbourg
Evelyne Lutton
Jacek Mañdziuk
Elena Marchiori
Dirk Mattfeld
Barry McCollum
Nicholas F. McPhee
Jörn Mehnen
Juan J. Merelo Guervós
Peter Merz
Silja Meyer-Nieberg
Zbigniew Michalewicz
Ralf Mikut
Boris S. Mitavskiy
Alberto Moraglio
Katharina Morik
Tomoharu Nakashima
Boris Naujoks
Frank Neumann
Shigeru Obayashi
Pietro S. Oliveto
Ben Paechter
Luís Paquete
Mario Pavone
Martin Pelikan
Mike Preuss
Christian Prins
Domenico Quagliarella
Günther Raidl
Robert G. Reynolds
Andrea Roli
Jonathan E. Rowe
Thomas P. Runarsson
Thomas A. Runkler
Conor Ryan
Michael Sampels
Jayshree Sarma
Robert Scheffermann
Lothar M. Schmitt
Marc Schoenauer
Oliver Schütze
Michèle Sebag
Bernhard Sendhoff
Marc Sevaux
Jonathan L. Shapiro

Ofer M. Shir
Joachim Sprave
Dipti Srinivasan
Thomas Stibor
Thomas Stützle
Dirk Sudholt
Ponnuthurai Suganthan
El-Ghazali Talbi
Kay Chen Tan
Alexander O. Tarakanov
Andrea G.B. Tettamanzi
Olivier Teytaud
Lothar Thiele
Dirk Thierens
Jonathan Timmis
Julian Togelius
Marco Tomassini
Heike Trautmann
Andrew Tuson
L. Gwenn Volkert
Michael D. Vose
Lipo Wang
Ingo Wegener
Lyndon While
L. Darrell Whitley
R. Paul Wiegand
Sławomir T. Wierzchoń
Carsten Witt
Xin Yao
Gary Yen
G. Tina Yu
Byoung-Tak Zhang
Qingfu Zhang
Karin Zielinski
Eckart Zitzler

Sponsoring Institutions

Sonderforschungsbereich ‘Computational Intelligence’ (SFB 531), Germany
Deutsche Forschungsgemeinschaft (DFG), Germany
Gesellschaft für Informatik (GI), Germany

Table of Contents

New Techniques

On the Behaviour of the (1+1)-ES for a Simple Constrained Problem

Dirk V. Arnold and Daniel Brauer

Faculty of Computer Science, Dalhousie University
Halifax, Nova Scotia, Canada B3H 1W5
{dirk,brauer}@cs.dal.ca

Abstract. This paper studies the behaviour of the (1 + 1)-ES when applied to a linear problem with a single linear constraint. It goes beyond previous work by considering constraint planes that do not contain the gradient direction. The behaviour of the distance of the search point from the constraint plane forms a Markov chain. The limit distribution of that chain is approximated using an exponential model, and progress rates and success probabilities are derived. Consequences for the working of step length adaptation mechanisms based on success probabilities are discussed.

1 Introduction

Constraint handling is an important aspect of numerical optimisation. See [6, 7, 10, 11, 14] and the references therein for examples of constraint handling techniques that have been proposed in connection with evolution strategies. The performance of new techniques is commonly evaluated using large and diverse sets of test functions, such as the benchmark set compiled for the *CEC 2006 Special Session on Constrained Real-Parameter Optimization* [8]. Moreover, the evaluation criteria used are relatively complex and involve various parameters, such as the number of function evaluations allowed and different quality thresholds. As a result, the observed outcomes are not always easy to interpret.

In contrast, in the realm of unconstrained optimisation there is a significant body of work employing simple test functions that aims at arriving at a better understanding of the behaviour of evolution strategies. See [1, 2, 5, 13] for examples and further references. Starting from the simplest non-trivial strategies and optimisation environments, the complexity of the scenarios studied has increased over time, and today results are available for adaptive strategies and problems with various degrees of ill-conditioning. The approach complements observations for large and difficult test beds with results that are easy to interpret, and that reveal scaling properties and the influence of parameters on optimisation performance.

The *Handbook of Evolutionary Computation* [3, page B2.4:**11**f] lists a small number of studies that use simple test functions and derive analytical results in the realm of constrained optimisation with evolution strategies. Rechenberg [12] studies the performance of the $(1+1)$-ES for the axis-aligned corridor model. Schwefel [15] considers the performance of the $(1, \lambda)$-ES in the same environment. Beyer [4] analyses the performance of the $(1+1)$-ES for a constrained function he refers to as a "discus". All of those have in common that the constraint planes are oriented such that they contain the gradient vector of the objective function. The goal of this paper is to take a step toward understanding the behaviour of evolution strategies in environments where there are constraints that do not contain the gradient direction. Specifically, we investigate the performance of the $(1+1)$-ES for a linear problem with a single linear constraint.

An issue of particular significance in the context of real-valued evolutionary optimisation is that of step length adaptation. In order to achieve good performance, the mutation strength of an evolution strategy needs to be adapted in the course of the search. For a linear problem with a single linear constraint, the optimal long-term strategy is to increase the step length of the algorithm. The mutation strength of the $(1+1)$-ES is typically adapted using the 1/5th success rule [12]. Without considering this quantitatively, Schwefel [16, page 116f] points out that using that rule, the presence of constraints may lead to the step length being reduced in situations where the angle between the gradient direction and the normal vector of the active constraint is small, leading to convergence to a non-stationary point.

The remainder of this paper is organised as follows. Section 2 describes the dynamical system formed by the operation of the $(1+1)$-ES for a linear problem with a single linear constraint using Markov chain terminology. A balance condition is derived that determines the limit distribution of that chain. Section 3 uses two approaches for obtaining an approximation to the limit distribution, and it evaluates them by comparing the quality of the derived predictions with measurements from runs of the strategy. Section 4 addresses the performance of step length adaptation based on success probabilities. Finally, Section 5 concludes with a brief discussion and suggestions for future work.

2 Dynamical System

Throughout this paper, we consider the problem of maximising[1] a linear function $f : \mathbb{R}^N \to \mathbb{R}$, $N \geq 2$, with a single linear constraint. We assume that the gradient vector of the objective function does not lie in the constraint plane. Without loss of generality, we choose a Euclidean coordinate system with its origin located on the constraint plane, and with its axes oriented such that the x_1-axis coincides with the gradient direction ∇f, and the x_2-axis lies in the two-dimensional plane spanned by the gradient vector and the normal vector of the constraint plane. The angle between those two vectors is denoted by θ as illustrated in Fig. 1, and it is referred to as the constraint angle. Constraint angles of interest are in $(0, \pi/2)$. The unit normal vector of the constraint plane expressed in the chosen coordinate system is $\mathbf{n} = \langle \cos\theta, \sin\theta, 0, \ldots, 0\rangle$. The signed

[1] Strictly speaking, the task is one of amelioration rather than maximisation, as a finite maximum does not exist. We do not make that distinction here.

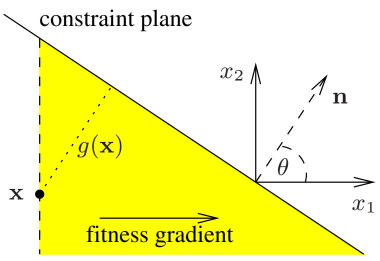


Fig. 1. Linear objective function with a single linear constraint. The subspace spanned by the x_1- and x_2-axes is shown. The search point $\mathbf{x}$ of the $(1+1)$-ES is at a distance $g(\mathbf{x})$ from the constraint plane. The shaded region consists of those points that are feasible and not inferior to the search point in terms of their objective function values.

distance of a point $\mathbf{x} = \langle x_1, x_2, \ldots, x_N \rangle \in \mathbb{R}^N$ from the constraint plane is thus $g(\mathbf{x}) = -\mathbf{n} \cdot \mathbf{x} = -x_1 \cos\theta - x_2 \sin\theta$, resulting in the optimisation problem

$$\text{maximise} \quad f(\mathbf{x}) = cx_1 \quad \text{subject to} \quad g(\mathbf{x}) \geq 0.$$

Notice that due to the choice of coordinate system, variables $x_3, x_4, \ldots, x_N$ enter neither the objective function nor the constraint inequality.

Given a feasible initial candidate solution $\mathbf{x}^{(0)}$, the $(1+1)$-ES generates a sequence $\mathbf{x}^{(t)}$, $t > 0$, of further feasible candidate solutions — sometimes referred to as search points — until some stopping criterion is satisfied. In time step t, offspring candidate solution $\mathbf{y}^{(t)} \in \mathbb{R}^N$ is generated by sampling an N-dimensional normal distribution with mean $\mathbf{x}^{(t)}$ and with covariance matrix $\sigma^2\mathbf{I}$, where $\mathbf{I}$ is the $N \times N$ identity matrix. Parameter σ is referred to as the mutation strength and determines the step length of the strategy. Until Section 4, it is assumed to be constant. Vector $\mathbf{z}^{(t)} = (\mathbf{y}^{(t)} - \mathbf{x}^{(t)})/\sigma$ is referred to as mutation vector. Candidate solution $\mathbf{x}^{(t+1)}$ equals $\mathbf{y}^{(t)}$ if $f(\mathbf{y}^{(t)}) \geq f(\mathbf{x}^{(t)})$ and $g(\mathbf{y}^{(t)}) \geq 0$; it equals $\mathbf{x}^{(t)}$ otherwise.

The probability of accepting a newly generated offspring candidate solution as well as the expected improvement in objective function value in time step t depend on the distance $g(\mathbf{x}^{(t)})$ of the search point from the constraint plane. The evolution of the normalised distance $\delta^{(t)} = g(\mathbf{x}^{(t)})/\sigma$ is described by

$$\delta^{(t+1)} = \begin{cases} \delta^{(t)} - z_1^{(t)} \cos\theta - z_2^{(t)} \sin\theta & \text{if } z_1^{(t)} \geq 0 \\ & \quad \text{and } \delta^{(t)} \geq z_1^{(t)} \cos\theta + z_2^{(t)} \sin\theta \\ \delta^{(t)} & \text{otherwise} \end{cases} \tag{1}$$

where $z_1^{(t)}$ and $z_2^{(t)}$ are the standard normally distributed components of mutation vector $\mathbf{z}^{(t)} = \langle z_1^{(t)}, z_2^{(t)}, \ldots, z_N^{(t)} \rangle$. Condition $z_1^{(t)} \geq 0$ holds iff $f(\mathbf{y}^{(t)}) \geq f(\mathbf{x}^{(t)})$; condition $\delta^{(t)} \geq z_1^{(t)} \cos\theta + z_2^{(t)} \sin\theta$ holds iff $\mathbf{y}^{(t)}$ is feasible. In what follows, time superscripts are omitted where possible without causing confusion.

Equation (1) describes a continuous-state Markov process. In order to derive conditions on its stationary limit distribution, consider random variable

$w = z_1 \cos\theta + z_2 \sin\theta$. Clearly, w is normally distributed with zero mean and unit variance. From the fact that z_2 is standard normally distributed, it follows that the likelihood function of z_1 given an observation of w is

$$L(z_1 = x \,|\, w = y) = \frac{1}{\sqrt{2\pi}\sin\theta} \exp\left(-\frac{1}{2}\left(\frac{y - x\cos\theta}{\sin\theta}\right)^2\right).$$

The posterior probability density of z_1 given an observation of w is proportional to the product of the prior density of z_1 and that likelihood function:

$$p(z_1 = x \,|\, w = y) = \frac{C}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}x^2} L(z_1 = x \,|\, w = y).$$

The normalising factor C can be obtained from the requirement that the integral

$$\frac{C}{2\pi\sin\theta}\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2}x^2} \exp\left(-\frac{1}{2}\left(\frac{y - x\cos\theta}{\sin\theta}\right)^2\right) \mathrm{d}x = \frac{C}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}y^2}$$

equal unity, yielding $C = \sqrt{2\pi}\mathrm{e}^{y^2/2}$. The posterior probability density is thus

$$\begin{aligned} p(z_1 = x \,|\, w = y) &= \frac{1}{\sqrt{2\pi}\sin\theta} \mathrm{e}^{\frac{1}{2}y^2} \mathrm{e}^{-\frac{1}{2}x^2} \exp\left(-\frac{1}{2}\left(\frac{y - x\cos\theta}{\sin\theta}\right)^2\right) \\ &= \frac{1}{\sqrt{2\pi}\sin\theta} \exp\left(-\frac{1}{2}\left(\frac{x - y\cos\theta}{\sin\theta}\right)^2\right). \end{aligned} \tag{2}$$

According to Eq. (1), the normalised distance δ of the search point from the constraint plane is updated by subtracting w if the result is nonnegative and $z_1 \geq 0$; it is unchanged otherwise. The probability that an update $w = y$ is accepted is thus zero if $\delta < y$; using Eq. (2), it equals

$$\begin{aligned} \mathrm{Prob}(z_1 \geq 0 \,|\, w = y) &= \int_0^{\infty} p(z_1 = x \,|\, w = y)\, \mathrm{d}x \\ &= \Phi\left(\frac{y}{\tan\theta}\right) \end{aligned} \tag{3}$$

where Φ denotes the cumulative distribution function of the standard normal distribution, if $\delta \geq y$.

The normalised distance $\delta^{(t+1)}$ of the search point from the constraint plane equals $\delta^{(t)}$ if the offspring candidate solution is rejected; it equals $\delta^{(t)} - w^{(t)}$ otherwise. Thus, $\delta^{(t+1)}$ is in an interval of width $\mathrm{d}x$ centred at x if either $\delta^{(t)}$ is in that interval and the offspring is rejected, or if $\delta^{(t)} - w^{(t)}$ is in that interval and the offspring candidate solution is accepted. The probability of the offspring candidate solution being accepted can be computed by integrating the probability described by Eq. (3), weighted with the probability density of generating the respective updates. Writing $p_\delta^{(t)}$ for the probability density of the normalised distance $\delta^{(t)}$ of the search point from the constraint plane at time t, it follows

$$p_\delta^{(t+1)}(x) = p_\delta^{(t)}(x)\left[1 - \frac{1}{\sqrt{2\pi}}\int_0^\infty \mathrm{e}^{-\frac{1}{2}(x-y)^2}\Phi\left(\frac{x-y}{\tan\theta}\right)\mathrm{d}y\right] + \frac{1}{\sqrt{2\pi}}\int_0^\infty p_\delta^{(t)}(y)\mathrm{e}^{-\frac{1}{2}(y-x)^2}\Phi\left(\frac{y-x}{\tan\theta}\right)\mathrm{d}y \tag{4}$$

for the probability density of $\delta^{(t+1)}$. The expression in square brackets is the probability that the offspring candidate solution is rejected. Equation (4) is the Chapman-Kolmogorov equation of the Markov process described by Eq. (1). A necessary condition for the stationary limit distribution is that it is stable under the update rule Eq. (1), i.e., that $p_\delta^{(t+1)} \equiv p_\delta^{(t)}$. From Eq. (4), it follows that

$$p_\delta(x)\int_0^\infty \mathrm{e}^{-\frac{1}{2}(x-y)^2}\Phi\left(\frac{x-y}{\tan\theta}\right)\mathrm{d}y = \int_0^\infty p_\delta(y)\mathrm{e}^{-\frac{1}{2}(y-x)^2}\Phi\left(\frac{y-x}{\tan\theta}\right)\mathrm{d}y \tag{5}$$

needs to hold for all $x \geq 0$ in order for the distribution with density p_δ to be stable.

Macroscopic quantities of interest that can be computed if the limit distribution is known include the average normalised distance

$$\delta_{\mathrm{avg}} = \int_0^\infty x p_\delta(x)\,\mathrm{d}x \tag{6}$$

of the search point from the constraint plane. Similarly, the success probability, i.e., the probability that an offspring candidate solution replaces its parent, can be obtained by computing the expected value of the probability in Eq. (3):

$$P_{\mathrm{succ}} = \frac{1}{\sqrt{2\pi}}\int_0^\infty p_\delta(x)\int_0^\infty \mathrm{e}^{-\frac{1}{2}(x-y)^2}\Phi\left(\frac{x-y}{\tan\theta}\right)\mathrm{d}y\,\mathrm{d}x. \tag{7}$$

Finally, the progress rate, i.e., the expected distance that the search point progresses in the direction of the gradient of the objective function per time step, can be computed as

$$\varphi = \frac{1}{\sqrt{2\pi}}\int_0^\infty p_\delta(x)\int_0^\infty y\mathrm{e}^{-\frac{1}{2}y^2}\Phi\left(\frac{x-y\cos\theta}{\sin\theta}\right)\mathrm{d}y\,\mathrm{d}x \tag{8}$$

where $\Phi((x - y\cos\theta)/\sin\theta)$ equals the probability that an offspring candidate solution with $z_1 = y$ is feasible given that $\delta = x$.

3 Modelling the Limit Distribution

Unfortunately, it is not clear whether a closed form solution for the limit distribution exists. Instead, this section uses and compares two simple approaches in an attempt to arrive at a distribution that approximates the limit distribution.

3.1 Dirac Delta Model

As a zeroth order approximation, let us assume that the limit distribution is well characterised by its expectation, and that higher order moments can be neglected. That is, we model the limit distribution as a (shifted) Dirac delta function, and we obtain the mode of that distribution by requiring that

$$\mathrm{E}\left[\delta^{(t+1)}\right] = \delta^{(t)}. \tag{9}$$

Using Eq. (4) to compute the expected value of $\delta^{(t+1)}$ yields

$$\begin{aligned}\mathrm{E}\left[\delta^{(t+1)}\right] &= \int_0^\infty x p_\delta^{(t+1)}(x)\,\mathrm{d}x \\ &= \delta^{(t)} - \frac{1}{\sqrt{2\pi}}\left[\Phi\left(\frac{\delta^{(t)}}{\sin\theta}\right)\cos\theta - \mathrm{e}^{-\frac{1}{2}\delta^{(t)^2}}\Phi\left(\frac{\delta^{(t)}}{\tan\theta}\right)\right].\end{aligned} \tag{10}$$

Together, Eqs. (9) and (10) can be used to solve numerically for $\delta^{(t)}$ as the mode of the delta distribution. Equations (7) and (8) yield the corresponding success probability and progress rate. Figure 2 shows a comparison of the resulting predictions with measurements made in runs of the $(1+1)$-ES. Each data point has been obtained by running the evolution strategy, initialised at $\delta = 1$, for 1,000 time steps in order to attain a distribution of distances from the constraint plane that resembles the limit distribution. Then, distances from the constraint plane, the number of successful mutations, and the progress in direction of the x_1-axis are measured for a further 10^6 time steps to yield the data points shown. It can be seen that the agreement of the measured data points with the predictions made on the basis of the delta model is not very good.

The approach of modelling unknown distributions using a delta distribution and requiring that the expected value of a state variable remain unchanged under the update rule of the system has been used extensively by Beyer [5]. It has been employed in a variety of situations where the variance of the limit distribution tends to zero as the search space dimensionality increases, and where the approach becomes increasingly exact for large N. Clearly, this is not the case in the present situation. In order to obtain improved approximations in finite-dimensional search spaces, Beyer [5] proposes modelling the limit distributions of the state variables using the first $k \geq 1$ terms of their Gram-Charlier expansions, and to determine the unknown k moments by computing their values after a time step and requiring that they equal their corresponding values before the time step. The use of the delta model as described above is the application of that approach for $k = 1$. While better approximations can potentially be achieved for larger values of k, we do not pursue this approach here as it is unclear whether the limit distribution is well described by any small number of lower order moments.

3.2 Exponential Model

Instead, motivated by the visual inspection of measured densities for several constraint angles θ, we choose to model the limit distribution of the normalised

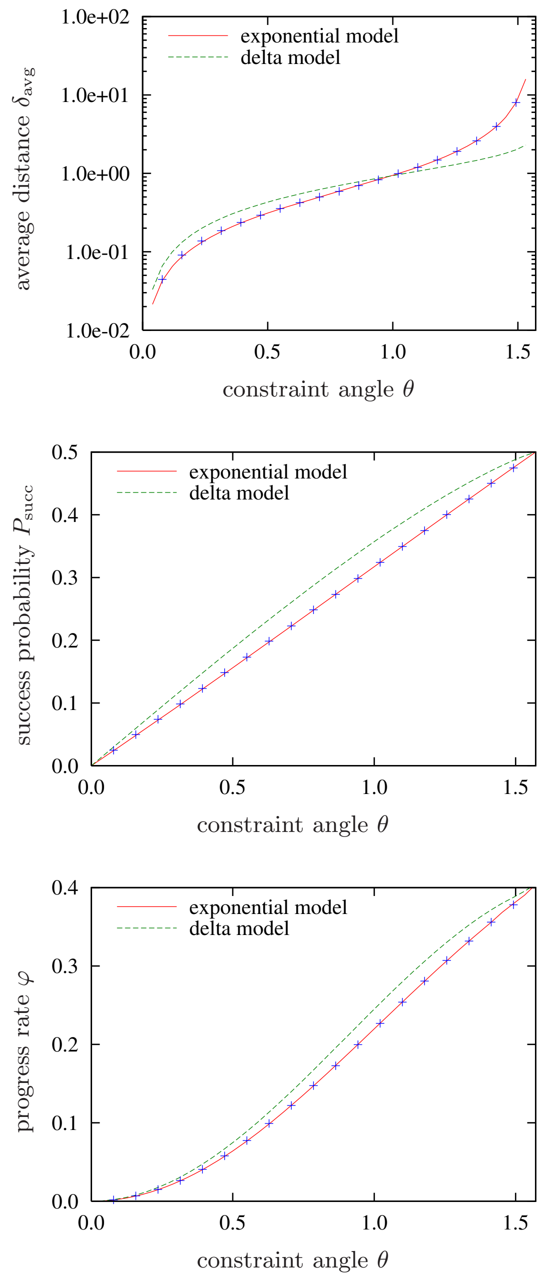


Fig. 2. Average distance δ_{avg} of the search point from the constraint plane, success probability P_{succ}, and progress rate φ plotted against the constraint angle θ. The dashed and solid lines represent predictions using the delta and exponential models from Sections 3.1 and 3.2, respectively. The dots mark measurements from runs of the evolution strategy.

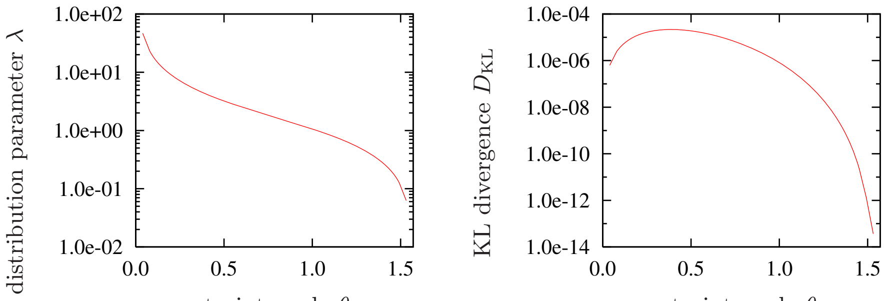


Fig. 3. Parameter λ of the exponential distribution that minimises the Kullback-Leibler divergence D_{KL} and that divergence plotted against the constraint angle θ

distance of the search point from the constraint plane using an exponential distribution with density $p(x) = \lambda \mathrm{e}^{-\lambda x}$. The distribution parameter λ is determined by minimising the Kullback-Leibler divergence

$$D_{\mathrm{KL}} = \int_0^\infty p_\delta^{(t)}(x) \log \frac{p_\delta^{(t)}(x)}{p_\delta^{(t+1)}(x)} \,\mathrm{d}x$$

of the distribution of $\delta^{(t+1)}$ from that of $\delta^{(t)}$. Using the exponential model for $p_\delta^{(t)}$ and Eq. (4) for $p_\delta^{(t+1)}$, it follows after rearranging terms that

$$\begin{aligned} D_{\mathrm{KL}} = &- \int_0^\infty \lambda \mathrm{e}^{-\lambda x} \\ &\cdot \log\left(1 - \frac{1}{\sqrt{2\pi}} \int_0^\infty \mathrm{e}^{-\frac{1}{2}(x-y)^2} \left[1 - \Phi\left(\frac{y-x}{\tan\theta}\right)\left(1 + \mathrm{e}^{-\lambda(y-x)}\right)\right] \mathrm{d}y\right) \mathrm{d}x. \end{aligned}$$

Solving for λ by minimising numerically and using the outcome in Eqs. (6), (7), and (8) yields the results shown in Fig. 2 as solid curves. The agreement with the values measured in runs of the $(1+1)$-ES is much better than for the delta model from Section 3.1. The parameter λ of the exponential distribution that minimises the Kullback-Leibler divergence along with the corresponding value of the latter are shown in Fig. 3.

4 Step Length Adaptation

The mutation strength σ, which controls the step length of the $(1+1)$-ES, is typically adapted based on the fraction of the most recently generated offspring candidate solutions that have been accepted. If that fraction is small, the mutation strength is reduced; if the fraction is large, the mutation strength is increased. Based on investigations of two test functions, Rechenberg [12] suggests that the success probability should ideally be in the vicinity of 20% ("1/5th success rule").

As pointed out by Schwefel [16], adaptation of the mutation strength based on success probabilities fails if the angle between the gradient vector of the objective function and the normal vector of the constraint plane is too small. It can be seen from Fig. 2 that for a given mutation strength, success probabilities in the limit state are below 20% if $\theta < 0.634$. For smaller constraint angles the step length is reduced systematically and the strategy converges to a non-stationary point. For larger constraint angles the success probability exceeds 20% and the mutation strength is increased indefinitely (which is useful in the long term, as the progress rate of the strategy increases with the mutation strength). Choosing a different target success probability shifts the point where the breakdown of step length control occurs, but it does not solve the problem.

Not shown here, experiments with a randomised, simulated annealing inspired acceptance rule suggest that convergence to a non-stationary point can be avoided if inferior candidate solutions are accepted with a non-zero probability. This is reminiscent of recent findings by Meyer-Nieberg and Beyer [9] who show that noise can improve the performance of an evolution strategy employing mutative self-adaptation for the sharp ridge function by preventing it from approaching the ridge too closely. Understanding the effects of randomised selection as well as the exact behaviour of the strategy near the point where the breakdown occurs remain as tasks for future work.

5 Summary and Discussion

To conclude, in this paper we have studied the behaviour of the $(1+1)$-ES for a linear problem with a single linear constraint. For constant mutation strength, the distance of the search point from the constraint plane is the state variable of a one-dimensional Markov process. Approximations to the limit distribution of that process have been obtained using two different models, and macroscopic quantities such as the success probability and the progress rate of the strategy have been derived.

The interaction between the step length adaptation mechanism of an evolution strategy and its approach to handling constraints is of crucial importance for the strategy's potential to solve constrained optimisation problems. It is not immediately obvious that all of the combinations of constraint handling and step length adaptation techniques that can be found in the literature are capable of achieving good performance in an environment as simple as the one considered here, and we would argue that a linear problem with a single linear constraint, along with other simple functions, such as linearly constrained spheres, may be a useful test case for real-valued evolutionary algorithms for constrained optimisation.

Clearly, this paper is but one step toward the goal of improving the understanding of constraint handling techniques in evolutionary computation by investigating their behaviour for simple test functions. In future work, we plan to derive formal convergence criteria that allow us to establish conditions on the existence of stable limit distributions, both for the adaptive and non-adaptive cases. It is also desirable to consider more complex types of evolution strategies that employ different step length adaptation mechanisms, different constraint handling techniques, and further constrained test functions.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Arnold, D.V.: On the use of evolution strategies for optimising certain positive definite quadratic forms. In: Proceedings of the 2007 Genetic and Evolutionary Computation Conference — GECCO 2007, pp. 634–641. ACM Press, New York (2007)
[2] Arnold, D.V., MacLeod, A.: Step length adaptation on ridge functions. Evolutionary Computation 16(2), 151–184 (2008)
[3] Bäck, T., Fogel, D.B., Michalewicz, Z.: Handbook of Evolutionary Computation. Oxford University Press, Oxford (1997)
[4] Beyer, H.-G.: Ein Evolutionsverfahren zur mathematischen Modellierung stationärer Zustände in dynamischen Systemen. PhD thesis, Hochschule für Architektur und Bauwesen, Weimar (1989)
[5] Beyer, H.-G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001)
[6] Coello Coello, C.A.: Constraint-handling techniques used with evolutionary algorithms. In: Proceedings of the 2007 Genetic and Evolutionary Computation Conference — GECCO 2007, pp. 3057–3077. ACM Press, New York (2007)
[7] Kramer, O., Schwefel, H.-P.: On three new approaches to handle constraints within evolution strategies. Natural Computing 5(4), 363–385 (2006)
[8] Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello Coello, C.A., Deb, K.: Problem definitions and evaluation criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore (2006)
[9] Meyer-Nieberg, S., Beyer, H.-G.: Why noise be good: Additive noise on the sharp ridge. In: Proceedings of the 2008 Genetic and Evolutionary Computation Conference — GECCO 2008, may, ACM Press, New York (to appear, 2008)
[10] Montes, E.M., Coello Coello, C.A.: A simple multi-membered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation 9(1), 1–17 (2005)
[11] Oyman, A.I., Deb, K., Beyer, H.-G.: An alternative constraint handling method for evolution strategies. In: Proc. of the 1999 IEEE Congress on Evolutionary Computation, pp. 612–619. IEEE Computer Society Press, Los Alamitos (1999)
[12] Rechenberg, I.: Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Friedrich Frommann Verlag (1973)
[13] Rechenberg, I.: Evolutionsstrategie ’94. Friedrich Frommann Verlag (1994)
[14] Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation 4(3), 274–283 (2000)
[15] Schwefel, H.-P.: Numerical Optimization of Computer Models. Wiley, Chichester (1981)
[16] Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley, Chichester (1995)

σ-Self-Adaptive Weighted Multirecombination Evolution Strategy with Scaled Weights on the Noisy Sphere

Hans-Georg Beyer and Alexander Melkozerov

Research Center Process and Product Engineering
Department of Computer Science
Vorarlberg University of Applied Sciences
Hochschulstr. 1, A-6850 Dornbirn, Austria
{hans-georg.beyer,alexander.melkozerov}@fhv.at

Abstract. This paper presents a performance analysis of the recently proposed σ-self-adaptive weighted recombination evolution strategy (ES) with scaled weights. The steady state behavior of this ES is investigated for the non-noisy and noisy case, and formulas for the optimal choice of the learning parameter are derived allowing the strategy to reach maximal performance. A comparison between weighted multirecombination ES with σ-self-adaptation (σSA) and with cumulative step size adaptation (CSA) shows that the self-adaptive ES is able to reach similar (or even better) performance as its CSA counterpart on the noisy sphere.

1 Introduction

The σ-self-adaptive weighted multirecombination ES, short $(\lambda)_{\text{opt}}$-σSA-ES, has been proposed in [1] as a new ES which takes advantage of Arnold's weighted multirecombination [2] and σ-self-adaptation [3] at the same time. It has been shown, considering the sphere model, that the $(\lambda)_{\text{opt}}$-σSA-ES can outperform Arnold's $(\lambda)_{\text{opt}}$-ES with cumulative step size adaptation (CSA) which was regarded as the most efficient ES with isotropic mutations [1].

While this performance advantage of the new $(\lambda)_{\text{opt}}$-σSA-ES is rather small, the remarkable aspect of this finding concerns the possible implications for more advanced ES which rely on correlated mutations, such as the CMA-ES [4,5]. It seems that the general concept of (mutative) self-adaptation can be transfered to algorithms, which rely on covariance matrix information without severe performance degradation. While a proof of this statement must be deferred to another paper (see [6] in this volume), the properties of the $(\lambda)_{\text{opt}}$-σSA-ES on simple fitness environments have not been sufficiently investigated up until now. However, understanding the behavior of the new ES on simple test scenarios can be regarded as a necessary building block for understanding and designing more complex algorithms which rely on correlated mutations.

In this paper, we investigate the behavior of the $(\lambda)_{\text{opt}}$-σSA-ES with *scaled* weights on the noisy sphere. Usage of scaled weights allows for larger optimal mutation strengths [2]. It can be beneficial in noisy fitness environments as the ES will work with larger mutations [7], although strongly scaled weights can deteriorate the maximal attainable progress rate in the non-noisy case (in finite-dimensional search spaces). Besides the performance analysis of the $(\lambda)_{\mathrm{opt}}$-σSA-ES under constant non-normalized noise strength conditions, we will also present a comparison with the $(\lambda)_{\mathrm{opt}}$-CSA-ES.

The paper is organized as follows. Section 1.1 provides the description of the $(\lambda)_{\mathrm{opt}}$-σSA-ES algorithm. Section 1.2 presents the derivation of the formula for optimal learning parameter. Section 2 is devoted to the analysis of the behavior in the constant non-normalized noise scenario. Part of this investigation is a comparison of the $(\lambda)_{\mathrm{opt}}$-σSA-ES with the $(\lambda)_{\mathrm{opt}}$-CSA-ES on the noisy sphere in Section 2.2. Section 3 summarizes the results obtained and draws conclusions.

1.1 The $(\lambda)_{\mathrm{opt}}$-σ-Self-adaptation-ES

The weighted multirecombination ES with σ-self-adaptation is a result of the combination of the σ-self-adaptation technique with the $(\lambda)_{\mathrm{opt}}$-ES. The algorithm of the $(\lambda)_{\mathrm{opt}}$-σ-self-adaptation-ES is given below [1]:

```
1   $\sigma_{\mathrm{p}} \leftarrow \sigma_{\mathrm{init}}$
2   $\mathbf{y}_{\mathrm{p}} \leftarrow \mathbf{y}_{\mathrm{init}}$
3   do
4     for $l = 1$ to $\lambda$
5       $\tilde{\sigma}_l \leftarrow \sigma_{\mathrm{p}} \mathrm{e}^{\tau \mathcal{N}_l(0,1)}$
6       $\tilde{\mathbf{z}}_l \leftarrow \boldsymbol{\mathcal{N}}_l(\mathbf{0}, \mathbf{I})$
7       $\tilde{\mathbf{y}}_l \leftarrow \mathbf{y}_{\mathrm{p}} + \tilde{\sigma}_l \tilde{\mathbf{z}}_l$
8       $\tilde{f}_l \leftarrow f(\tilde{\mathbf{y}}_l)$
9     end
10    $\langle \sigma \rangle \leftarrow \frac{1}{\mu} \sum_{m=1}^{\mu} \tilde{\sigma}_{m;\lambda}$
11    $\langle \mathbf{z} \rangle_\omega \leftarrow \sum_{l=1}^{\lambda} \omega_{l;\lambda} \tilde{\mathbf{z}}_{l;\lambda}$
12    $\sigma_{\mathrm{p}} \leftarrow \langle \sigma \rangle$
13    $\mathbf{y}_{\mathrm{p}} \leftarrow \mathbf{y}_{\mathrm{p}} + \langle \sigma \rangle \langle \mathbf{z} \rangle_\omega$
14  until termination criterion fulfilled
```

Fig. 1. The pseudocode of the $(\lambda)_{\mathrm{opt}}$-σ-self-adaptation-ES

In the algorithm, the parent state is initialized in lines 1 and 2. λ offspring are generated from line 4 to line 9 in the following way: For each offspring, the mutation of the mutation strength is performed in line 5 using the log-normal operator $\mathrm{e}^{\tau \mathcal{N}_l(0,1)}$, where $\mathcal{N}_l(0,1)$ is a $(0,1)$ normally distributed random scalar. The learning parameter τ in the log-normal operator controls the self-adaptation rate. In line 6, direction of the mutation vector $\tilde{\sigma}_l \tilde{\mathbf{z}}_l$ is determined by means of a $(0,1)$ normally distributed random vector $\boldsymbol{\mathcal{N}}_l(\mathbf{0}, \mathbf{I})$. The offspring vector $\tilde{\mathbf{y}}_l$ is generated in line 7 and used in the calculation of the objective function value $\tilde{f}_l$ in line 8.

After creation, the λ offspring are ranked according to their objective function values and intermediate recombination of mutation strengths is performed in line 10 using the μ best individuals. In line 11, the weighted sum $\langle \mathbf{z} \rangle_\omega$ of mutation vectors is calculated w.r.t. the fitness of each offspring. The superscript $(l;\lambda)$ refers to the lth-best of the λ offspring (the lth-smallest for minimization). Weights $\omega_{l,\lambda}$ are dependent on the rank of the individual in the set of all offspring individuals [2]. The new parent state is obtained in lines 12 and 13.

After the termination criterion is fulfilled, the current parent state is considered as an approximation of the optimizer of the objective function $f(\mathbf{y})$.

1.2 Performance Analysis of the $(\lambda)_{\text{opt}}$-σSA-ES

The analysis of the $(\lambda)_{\text{opt}}$-σSA-ES is done for the noisy quadratic sphere[1] with search space dimensionality N. Due to space limitation, we can only sketch the derivation steps (see also [1]). Since the self-adaptation mechanism from the standard $(\mu/\mu_I,\lambda)$-σSA-ES is introduced in the $(\lambda)_{\text{opt}}$-σSA-ES without any changes, the following approximate formula for the self-adaptation response (SAR) of the $(\mu/\mu_I,\lambda)$-σSA-ES obtained for $N \to \infty$ and $\tau \ll 1$ in [8] can be applied to the analysis of the $(\lambda)_{\text{opt}}$-σSA-ES

$$\psi(\sigma^{*(g)}, \sigma_\epsilon^{*(g)}) \approx \tau^2 \left(\frac{1}{2} + \frac{(\sigma^{*(g)})^2}{(\sigma^{*(g)})^2 + (\sigma_\epsilon^{*(g)})^2} e^{1,1}_{\mu,\lambda} - \frac{(\sigma^{*(g)})^2}{\sqrt{(\sigma^{*(g)})^2 + (\sigma_\epsilon^{*(g)})^2}} c_{\mu/\mu,\lambda} \right), \tag{1}$$

where $e^{a,b}_{\mu,\lambda}$ is the generalized progress coefficient

$$e^{a,b}_{\mu,\lambda} = \frac{\lambda-\mu}{\sqrt{2\pi}^{a+1}} \binom{\lambda}{\mu} \int_{-\infty}^{\infty} t^b \mathrm{e}^{-\frac{a+1}{2}t^2} \Phi(t)^{\lambda-\mu-1} (1-\Phi(t))^{\mu-a} \mathrm{d}t, \tag{2}$$

with $\Phi(t)$ denoting the cumulative distribution function of the standard normal variate and $c_{\mu/\mu,\lambda}$ is the progress coefficient, $c_{\mu/\mu,\lambda} = e^{1,0}_{\mu,\lambda}$. The SAR function describes the expected parental normalized relative σ^*-change[2] given $\sigma^{*(g)} = \sigma^*_{\text{p}}$. Besides the dependencies on parent and offspring population size μ and λ, it also depends on the amount of (normalized) noise $\sigma^*_\epsilon = \sigma_\epsilon N/(2R^2)$.

In order to obtain maximal (normalized) quality gain Δ^* per generation (note, Δ is the expected fitness gain per generation), optimal weights $\omega_{l,\lambda}$ must be used. The following choice of scaled weights has been shown to be optimal in noisy fitness environments [2]

$$\omega_{l,\lambda} = E_{l,\lambda}/\kappa \quad \text{for } l = 1,\ldots,\lambda, \tag{3}$$

where $\kappa > 1$ and $E_{l,\lambda}$ is the expectation of the $(\lambda+1-l)$th order statistic of the standard normal variate, $E_{l,\lambda} = e^{0,1}_{l-1,\lambda}$. An asymptotically exact formula for normalized quality gain of the $(\lambda)_{\text{opt}}$-ES with the choice of weights (3) for

[1] The noisy quadratic sphere reads $f(\mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|^2 + \epsilon$, where $\hat{\mathbf{y}} \in \mathbb{R}^N$ is the optimizer and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$ is an additive normally distributed noise term.

[2] Note, $\sigma^* = \sigma N/R$, R is parental distance to the optimizer.

the noisy sphere model has been obtained in [2] using several simplifications (consideration of the asymptotic behavior for $N \to \infty$, assumption that the normalized mutation strength σ^* is of $\mathcal{O}(1)$, Taylor series expansion)

$$\Delta^*(\sigma^{*(g)}, \sigma_\epsilon^{*(g)}) = \frac{W_\lambda}{\kappa}\left(\frac{(\sigma^{*(g)})^2}{\sqrt{(\sigma^{*(g)})^2 + (\sigma_\epsilon^{*(g)})^2}} - \frac{(\sigma^{*(g)})^2}{2\kappa}\right), \tag{4}$$

where $W_\lambda = \sum_{l=1}^{\lambda} E_{l,\lambda}^2$.

Note, Eq. (4) has been derived for given values of $\sigma^{*(g)}$ and $\sigma_\epsilon^{*(g)}$, but, there are μ parents with different values. However, this causes no problems in the $N \to \infty$ limit [1]. The deviations vanish asymptotically. Therefore, one can use the normalized mean value $s^{*(g)} = \langle\sigma\rangle^*$ (i.e., the recombined σ of line 10 in Fig. 1) instead of individual $\sigma^{*(g)}$ in Eqs. (1) and (4). This simplifies the analysis considerably. Furthermore, in the limit case the normalized quality gain Δ^* becomes asymptotically equal to the normalized progress rate φ^* [9], i.e., $\Delta^*(s^{*(g)}) \simeq \varphi^*(s^{*(g)})$. The latter measures the expected distance gain per generation in the search space (note, this holds for the sphere model). Obviously, the choice of scaled weights (3) provides optimal progress rate as well.

The normalized mutation strength of an evolution strategy with correctly working σSA reaches a stationary state over time with $s_{\text{st}}^* = \lim_{g\to\infty} s^{*(g)}$, which is described by the steady state condition [10]

$$\frac{\varphi^*(s_{\text{st}}^{*(g)}, s_\epsilon^{*(g)})}{N} = -\psi\left(s_{\text{st}}^{*(g)}, s_\epsilon^{*(g)}\right). \tag{5}$$

This equation relates the expected distance change of the parental centroid (towards the optimizer) to the expected relative change of the mutation strength. It is therefore a fundamental equation for the analysis of self-adaptation and will be the basis for all considerations to come.

Let us first consider how to determine the optimal learning parameter τ in the non-noisy case $s_\epsilon^{*(g)} = 0$. Inserting (4) and (1) into (5) and using $\tau = \alpha/\sqrt{N}$ leads to

$$\frac{W_\lambda}{\kappa}\left(s_{\text{st}}^* - \frac{(s_{\text{st}}^*)^2}{2\kappa}\right) = -\alpha^2\left[\frac{1}{2} + e_{\mu,\lambda}^{1,1} - s_{\text{st}}^* c_{\mu/\mu,\lambda}\right]. \tag{6}$$

Note, the specific choice $\tau \propto \sqrt{N}$ has made (6) independent of N. Analytical solution of the quadratic equation (6) gives

$$s_{\text{st}}^* = \kappa\left[1 - \frac{\kappa c_{\mu/\mu,\lambda}\alpha^2}{W_\lambda} + K\right], \tag{7}$$

with $K = \sqrt{1 + (1 - 2\kappa c_{\mu/\mu,\lambda} + 2e_{\mu,\lambda}^{1,1})\frac{\alpha^2}{W_\lambda} + \frac{\kappa^2 c_{\mu/\mu,\lambda}^2 \alpha^4}{W_\lambda^2}}$.

If we take into account $\varphi^*(s^{*(g)}) = \Delta^*(s^{*(g)})$ and insert the stationary mutation strength (7) into the quality gain (4), we obtain the stationary progress rate as a function of α

$$\varphi^*_{\mathrm{st}}(\alpha) = \frac{W_\lambda}{2}\left[1 - \left(\frac{\kappa c_{\mu/\mu,\lambda}\alpha^2}{W_\lambda} - K\right)^2\right]. \tag{8}$$

One can easily check that the quality gain $\Delta^*(s^{*(g)}, 0)$ reaches its maximum $\Delta^*_{\max} = \frac{W_\lambda}{2}$ at $s^*_{\Delta_{\max}} = \kappa$. Therefore, obtaining maximal progress in the stationary state, one has to require $s^*_{\mathrm{st}} = s^*_{\Delta_{\max}} = \kappa$. Using (7) this leads to $\kappa\left[1 - \kappa c_{\mu/\mu,\lambda}\alpha^2_{\mathrm{opt}}/W_\lambda + K\right] = \kappa$ and after a short calculation, one gets

$$\alpha_{\mathrm{opt}} = \sqrt{\frac{W_\lambda}{2\kappa c_{\mu/\mu,\lambda} - 2e^{1,1}_{\mu,\lambda} - 1}}. \tag{9}$$

Formula (9) allows for calculating the optimal learning parameter τ (via $\tau_{\mathrm{opt}} = \alpha_{\mathrm{opt}}/\sqrt{N}$) corresponding to maximal progress rate in the non-noisy case. Note, we did not take into account the noise in this calculation because in practice the normalized noise strength is usually unknown. Instead, we will use (9) as a fixed value even when noise is present. Actually, this choice overestimates τ for non-vanishing noise strengths $s^{*(g)}_\epsilon$.

2 Constant Non-normalized Noise Strength

2.1 Determining the Residual Location Error

For constant non-normalized noise s_ϵ = const, the normalized noise strength increases while the ES approaches the optimizer $\hat{\mathbf{y}}$ (recall $\sigma^*_\epsilon = \sigma_\epsilon N/(2R^2)$ and R is the parental distance to the optimizer). Therefore, when the ES starts far away from the optimizer, the influence of the noise on the objective function values is small, but permanently increases with each step of the ES towards the optimum. Finally the ES enters a stationary state. The values of the objective function are completely shaded by the noise term ϵ and no further progress towards the optimum is possible. The distance to the optimizer R fluctuates around an expected value R_∞, which is referred to as *residual location error*. Furthermore, the mutation strength also converges to a stationary distribution.

The steady state behavior of the ES for constant non-normalized noise is described using the steady-state conditions (neglecting fluctuations)

$$R^{(g+1)} = R^{(g)} = R_\infty, \qquad s^{*(g+1)} = s^{*(g)} = s^*_{\mathrm{st}}, \tag{10}$$

which can also be written using progress rate and SAR functions as

$$\varphi\left(s^*_{\mathrm{st}}, s^*_{\epsilon\mathrm{st}}\right) = 0, \qquad \psi\left(s^*_{\mathrm{st}}, s^*_{\epsilon\mathrm{st}}\right) = 0. \tag{11}$$

Inserting Eq. (4) and Eq. (1) into (11), the resulting system of equations can be solved for steady state normalized mutation strength and normalized noise strength

$$s^*_{\mathrm{st}} = \kappa\sqrt{\frac{2}{2\kappa c_{\mu/\mu,\lambda} - e^{1,1}_{\mu,\lambda}}} \tag{12}$$

$$s^*_{\epsilon\mathrm{st}} = 2\kappa\sqrt{\frac{\frac{1}{2} + e^{1,1}_{\mu,\lambda} - 2\kappa c_{\mu/\mu,\lambda}}{e^{1,1}_{\mu,\lambda} - 2\kappa c_{\mu/\mu,\lambda}}}. \tag{13}$$

Taking into account the normalization $s^*_\epsilon = \frac{N}{2\left(r^{(g)}\right)^2} s_\epsilon$, the residual location error is obtained

$$R_\infty = \sqrt{N s_\epsilon/(4 s^*_{\epsilon\mathrm{st}})}. \tag{14}$$

Using (13) and (14), R_∞ can be rewritten as

$$R_\infty = \sqrt{\frac{N s_\epsilon}{4\kappa}\sqrt{\frac{e^{1,1}_{\mu,\lambda} - 2\kappa c_{\mu/\mu,\lambda}}{\frac{1}{2} + e^{1,1}_{\mu,\lambda} - 2\kappa c_{\mu/\mu,\lambda}}}}. \tag{15}$$

The predictions of (15) are compared with the results of experiments in Fig. 2. A start vector $\mathbf{y}^{(0)} = \mathbf{10}$, an initial mutation strength $\sigma^{(0)} = 1$ and a noise strength $\sigma_\epsilon = 1$ were used with $\mu = 4$ and offspring number $\lambda = 10$. The learning parameters τ were calculated using α_{opt} obtained by means of (9). The residual location error was obtained by collecting data starting from generation $g_0 = 100,000$ up to generation $g_{\max} = 200,000$.

The experimental results in Fig. 2 exhibit satisfactory agreements with residual location error values calculated using Eq. (15) even for low search space dimensionalities, although (15) was derived from Eq. (4) and Eq. (1) being approximations in the limit $N \to \infty$. The residual location error decreases with the increase

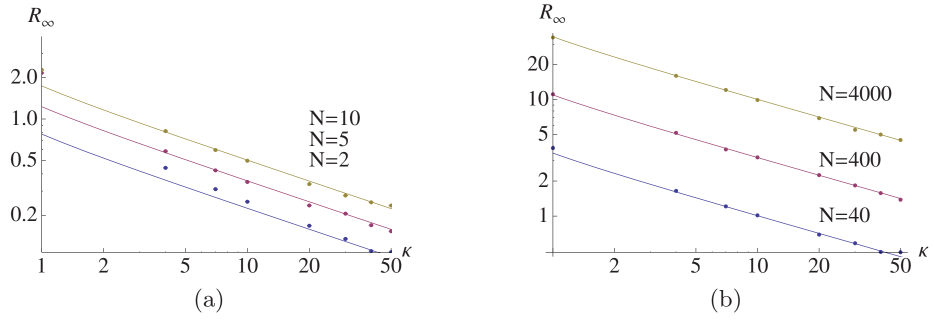


Fig. 2. The residual location error R_∞ of the $(\lambda)_{\mathrm{opt}}$-σSA-ES ($\mu = 4$, $\lambda = 10$) for different values of κ. The solid lines represent the results of (12) and (15). The points indicate the results of experiments for $N = 2, 5, 10$ from bottom to top in subfigure a) and for $N = 40, 400, 4000$ in subfigure b).

of κ. The reason is that the $(\lambda)_{\text{opt}}$-ES benefits from larger κ values by means of implicit rescaling of the mutation strength, allowing the ES to use higher mutation strengths in conjunction with smaller search point position changes [2].

2.2 Comparison of the $(\lambda)_{\text{opt}}$-σSA-ES with the $(\lambda)_{\text{opt}}$-CSA-ES

In order to compare the results of the $(\lambda)_{\text{opt}}$-σSA-ES for constant non-normalized noise strength with that of the $(\lambda)_{\text{opt}}$-CSA-ES, the same experiments were conducted for the $(\lambda)_{\text{opt}}$-CSA-ES and the number of offspring $\lambda = 10$ (Fig. 3). A start vector $\mathbf{y}^{(0)} = \mathbf{10}$, an initial mutation strength $\sigma^{(0)} = 1$ and a noise strength $\sigma_\epsilon = 1$ were used. The average quality gain for the $(\lambda)_{\text{opt}}$-CSA-ES is [2]

$$\Delta^*_{avg} = \begin{cases} \frac{\sqrt{2}-1}{2} W_\lambda \left(2 - \left(\frac{s^*_\epsilon}{\kappa}\right)^2\right) & \text{if } s^*_\epsilon < \sqrt{2}\kappa, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

The assumptions made for derivation of (16) in [2] do not allow us to directly obtain the stationary noise strength, two limit conditions are used to bracket its value:

1. Upper limit condition $s^*_\epsilon = \sqrt{2}\kappa$ due to Eq. (16). Using (14), one obtains the corresponding residual location error formula

$$R^{'}_\infty = \sqrt{N s_\epsilon / (2\sqrt{2}\kappa)}, \tag{17}$$

2. Lower limit condition $\varphi\left(s^*, s^*_\epsilon\right) = 0$, $s^* = 0$, which provides after resolving Eq. (4) for $s^*_{\epsilon\text{st}}$ and using (14) the residual location error formula

$$R_\infty = \sqrt{N s_\epsilon / (4\kappa)}. \tag{18}$$

Theoretical lines representing residual location error formulas for both limit conditions are shown in Fig. 3.

The experimental results for the $(\lambda)_{\text{opt}}$-CSA-ES are closer to the theoretical curve corresponding to the residual location error formula (18). This is an observation that cannot be deduced from the CSA theory developed so far.

Let us now consider the influence of λ on the R_∞-behavior of the strategies. As λ gets larger, using a fixed truncation ratio $0 < \mu/\lambda < 1$, the inner square root in (15) gets a constant. Furthermore, if κ is increased, this constant approaches 1, thus, yielding (18). That is, theoretically an influence of λ can only be introduced by choosing $\kappa = f(\lambda)$, e.g. $\kappa \propto \lambda$. Such investigations can be found in Fig. 4 for the $(\lambda)_{\text{opt}}$-σSA-ES using (15) and for the $(\lambda)_{\text{opt}}$-CSA-ES using (18) with $\kappa \propto \lambda$. The experimental results in Fig. 4 show that Eq. (15) provides satisfactory predictions of the residual location error R_∞ of the $(\lambda)_{\text{opt}}$-σSA-ES for a wide range of offspring numbers λ. The residual location error decreases with the increase of the number of offspring λ. The selection of the particular truncation ratio ($\mu/\lambda = 0.27$ or $\mu/\lambda = 0.4$) does not make a notable difference in the residual location error decrease rate.

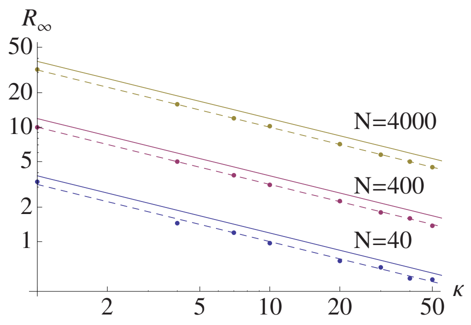


Fig. 3. The residual location error R_∞ of the $(\lambda)_{\text{opt}}$-CSA-ES ($\lambda = 10$) for different values of κ. The solid curves represent the results of (17), whereas dashed lines depict the results of (18). The points indicate the results of experiments for $N = 40, 400, 4000$ from bottom to top.

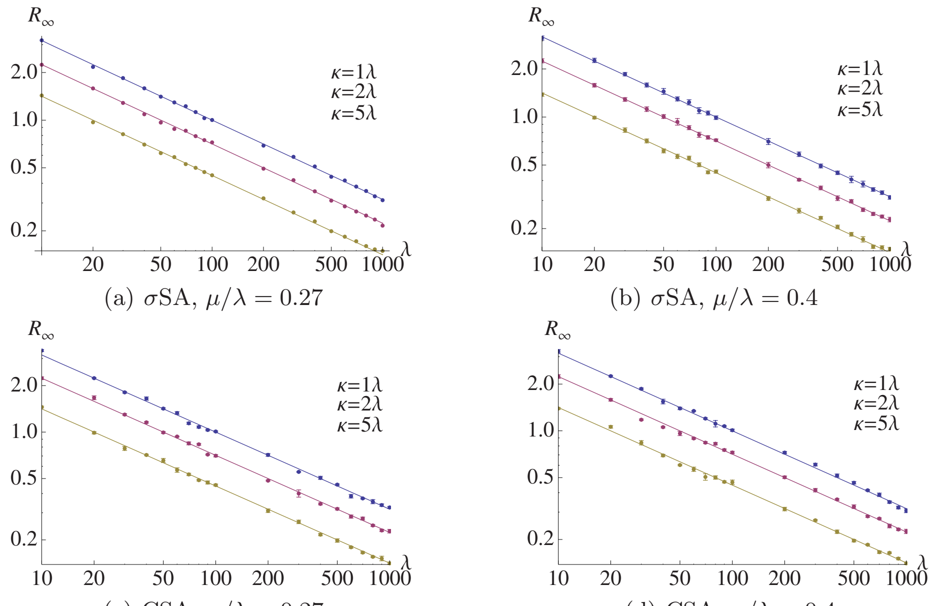


Fig. 4. The residual location error R_∞ of the $(\lambda)_{\text{opt}}$-σSA-ES and $(\lambda)_{\text{opt}}$-CSA-ES for different values of λ. The solid lines represent the results of (15) (σSA) and (18) (CSA) for $\kappa = \lambda$, $\kappa = 2\lambda$, $\kappa = 5\lambda$ from top to bottom. The points indicate the results of experiments for $N = 400$.

The comparison of the residual location error R_∞ of the $(\lambda)_{\text{opt}}$-CSA-ES for different values of λ with that of the $(\lambda)_{\text{opt}}$-σSA-ES in Fig. 4 shows that neither the CSA nor σ-self-adaptation have advantages over each other considering the task of reducing the residual location error by means of increasing the number of offspring λ.

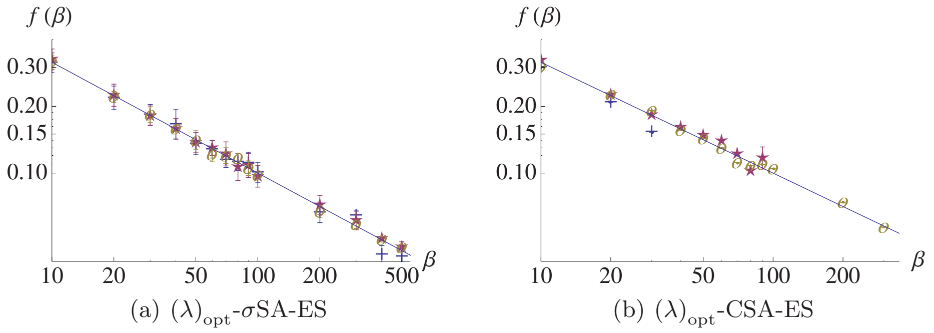


Fig. 5. The $f(\beta)$ plots for the $(\lambda)_{\text{opt}}$-σSA-ES and $(\lambda)_{\text{opt}}$-CSA-ES ($\mu = 4$, $\lambda = 10$). The solid curves represent the results of (15) with Eq. (19) applied. The points indicate the results of experiments for $N = 40$ (crosses), $N = 100$ (stars) and $N = 400$ (circles).

The κ-scaling behavior of both strategies in Fig. 2 and 3 makes an impression that choosing κ increasingly large one can get arbitrarily close to the optimizer without any additional costs (note, $\lambda = 10$). This observation might be due to the spherical symmetry of the sphere model. Investigations of other test functions are needed to clarify things. However, for the time being we want to check the limits of the theory developed. To this end, we introduce the scaling $\kappa = \beta\lambda$ ($\lambda = \text{const}$). If one inserts this in (18) and rearranges the expression, one obtains

$$\frac{2R_\infty\sqrt{\lambda}}{\sqrt{Ns_\epsilon}} = \frac{1}{\sqrt{\beta}} = f(\beta). \tag{19}$$

That is, the left hand side of (19) should appear as a linear decreasing curve in a double-logarithmic plot, which is independent of N (Fig. 5). Systematic deviations from this linear curve may signalize a break-down of the theory.

The results for the $(\lambda)_{\text{opt}}$-σSA-ES are presented in Fig. 5 up to $\beta = 500$ since this ES is capable of positive progress toward the optimum for very large β. In contrast, the results for the $(\lambda)_{\text{opt}}$-CSA-ES are presented up to $\beta < N$ since this ES exhibits divergent behavior for $\beta > N$ (no sharp bounds!). Therefore, the σ-self-adaptation is to a certain extent more robust for large κ values which allow for getting smaller R_∞, while CSA fails to adapt the mutation strength correctly in these extreme cases.

3 Summary and Conclusions

In this work, the behavior of the $(\lambda)_{\text{opt}}$-σSA-ES with scaled weights has been studied on the noisy sphere. At first, the stationary state condition of this ES has been investigated for the non-noisy case and the optimal learning parameter $\tau = \alpha_{\text{opt}}/\sqrt{N}$ has been obtained. The α_{opt} formula does not require information about noise strength, which is usually not known in real optimization tasks. This

α_{opt} can be used in practice, however, one should keep in mind that the ES's dynamical performance will not be optimal in noisy environments.

For constant non-normalized noise strength, theoretical formulas for steady state mutation strength and residual location error were derived and compared with experiments for different values of κ exhibiting satisfactory agreement with theoretical predictions. Residual location error plots for different values of λ were presented as well, showing that the particular truncation ratio ($\mu/\lambda = 0.27$ or $\mu/\lambda = 0.4$) does not make a notable difference in the residual location error decrease rate and that neither the CSA nor σSA have advantages over each other considering the task of reducing the residual location error by means of increasing of the number of offspring λ. However, keeping $\lambda = \text{const}$, the $(\lambda)_{\text{opt}}$-σSA-ES can reach smaller residual location errors than its CSA counterpart when using large κ values. Using very large κ values, the CSA fails to adapt the mutation strength correctly. The question whether this observation can be confirmed for other test functions with noise will be a direction of future investigations. Furthermore, the constant normalized noise case should be considered, too.

Acknowledgments. This work was supported by the Austrian Science Fund (FWF) under grant P19069-N18.

References

1. Beyer, H.-G., Melkozerov, A.: Mutative σ-Self-Adaptation Can Beat Cumulative Step Size Adaptation when Using Weighted Recombination. In: GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, Atlanta, GA, USA. ACM Press, New York (accepted, 2008)
2. Arnold, D.V.: Weighted Multirecombination Evolution Strategies. Theoretical Computer Science 361(1), 18–37 (2006)
3. Rechenberg, I.: Evolutionsstrategie 1994. Frommann-Holzboog, Stuttgart (1994)
4. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (2001)
5. Hansen, N., Müller, S., Koumoutsakos, P.: Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
6. Beyer, H.-G., Sendhoff, B.: Covariance Matrix Adaptation Revisited – the CMSA Evolution Strategy. In: Rudolph, G., et al. (eds.) Parallel Problem Solving from Nature 10. LNCS, vol. 5199. Springer, Heidelberg (2008)
7. Beyer, H.-G.: The Theory of Evolution Strategies. Natural Computing Series. Springer, Heidelberg (2001)
8. Meyer-Nieberg, S.: Self-Adaptation in Evolution Strategies. PhD Thesis, University of Dortmund, CS Department, Dortmund, Germany (2007)
9. Arnold, D.V., Beyer, H.-G.: Local Performace of the $(\mu/\mu_I, \lambda)$-ES in a Noisy Environment. In: Martin, W., Spears, W. (eds.) Foundations of Genetic Algorithms, vol. 6, pp. 127–141. Morgan Kaufmann, San Francisco (2001)
10. Meyer-Nieberg, S., Beyer, H.-G.: On the Analysis of Self-Adaptive Recombination Strategies: First Results. In: Proceedings of the CEC 2005 Conference, pp. 2341–2348. IEEE, Piscataway (2005)

Convergence Analysis of Evolution Strategies with Random Numbers of Offspring

Olivier François

Institut National Polytechnique de Grenoble,
TIMC-IMAG, Faculté de Médecine,
38706 La Tronche, France

Abstract. Hitting times of the global optimum for evolutionary algorithms are usually available for simple unimodal problems or for simplified algorithms. In discrete problems, the number of results that relate the convergence rate of evolution strategies to the geometry of the optimisation landscape is restricted to a few theoretical studies. This article introduces a variant of the canonical $(\mu + \lambda)$-ES, called the Poisson-ES, for which the number of offspring is not deterministic, but is instead sampled from a Poisson distribution with mean λ. After a slight change on the rank-based selection for the μ parents, and assuming that the number of offspring is small, we show that the convergence rate of the new algorithm is dependent on a geometric quantity that measures the maximal width of adaptive valleys. The argument of the proof is based on the analogy of the Poisson-ES with a basic Mutation-or-Selection evolutionary strategy introduced in a previous work.

Keywords: Evolution Strategies, Discrete Optimisation, Convergence Theory, Markov Chains, Large Deviations, Mutation or Selection.

1 Introduction

Evolution Strategies (ES) have generated considerable interest during the last decades, both in the practical and in the theoretical issues [3,5]. Until recently, however, the number of mathematical results about the behaviour of ES has remained rather limited, especially in the field of discrete optimisation. The early theoretical analyses indeed concentrated on continuous optimisation problems, and they were mainly based on the so-called rate-of-progress theory, examining the average gain of the algorithm after a single step of the algorithm [4]. In the continuous setting, global convergence results and results on hitting times of the global optimum are now at least available for simple unimodal problems like the sphere or quadratic functions [1,6], or for simplified algorithms like the $(1 + 1)$-ES [15].

Regarding discrete or combinatorial optimisation, the convergence analysis of evolutionary algorithms has also focused on simple cases, the most representative of which may be the *one-max* problem [9]. Numerous studies have obtained deep insights on such simple problems [18,19], like bounds for the runtime of simple

EA on pseudo-boolean functions [20,21]. Although some remarkable progress has been achieved for more complex problems [17], the transfer of results and techniques to new problems remain an open question. At the exception of the simulated annealing algorithm [14,19] and of a few variants of evolution strategies that were based on mutation or selection instead of mutation plus selection [10,12], few explicit results have linked hitting times of the global optimum to the geometrical features of the discrete optimisation landscape for an arbitrary optimisation problem. Nevertheless, these scarce results have revealed to be of fundamental interest as they have emphasised the importance of the depths of the adaptive valleys in the simulated annealing algorithm [14], and the importance of their widths in rank-based selection evolutionary algorithms [10].

One difficulty with building a convergence theory for discrete optimisation evolution strategies is the determinism of the selection schemes based on fitness rankings. In this article, we introduce a stochastic variant of ES that converts the usual deterministic offspring assumption made in these algorithms, into an assumption of a stochastic number of offspring. We show that this modification is crucial for characterising the convergence of evolution strategies by means of geometrical quantities.

The article is organised as follows. In section 2, we introduce the Poisson-Evolution Strategy (Poisson-ES) in which the number of offspring is randomly sampled according to the Poisson distribution with mean λ. This modification of the canonical ES will be accompanied by a slight change on the deterministic rank-based selection for the parents, which objective is to prevent premature convergence. Section 3 states our main results about hitting times of the global optimum that are valid for small λ. These results underline the role of the width of adaptive valleys for determining the rate of evolution toward the global optimum. Section 4 presents a short simulation study illustrating the fact that the theory can also predict the behaviour of the modified algorithm for values of λ that are not close to zero.

2 The Poisson-ES

Consider a finite set, V, and assume that we seek the maximum of an injective objective function f defined on V

$$f : \quad V \to \mathbb{R}_+ \,.$$

Here injectivity is more a convenient assumption than a necessary condition. It facilitates proofs and leads to more elegant statements (see [12] for a more general setting). The canonical $(\mu + \lambda)$-ES is usually defined as follows. At each generation, the algorithm generates a deterministic number of offspring, λ, from μ parents, and simultaneously applies a mutation operator to the λ offspring. Then, μ individuals are selected among the $(\mu + \lambda)$ parents plus offspring to form the parental population in the next generation.

Here, we introduce a variant of the $(\mu + \lambda)$-ES, that generates stochastic – in place of deterministic – numbers of offspring in each generation. In the variant under consideration, the number of offspring, Λ, is sampled from a Poisson distribution with mean λ, for some value $\lambda > 0$. The probability that the algorithm generates k offspring in a given generation is then equal to

$$\Pr(\text{generate } k \text{ offspring}) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k \geq 0\,.$$

One key property of the Poisson distribution regarding the further analysis of the stochastic dynamics of the algorithm is that

$$\Pr(\text{generate exactly 1 offspring}) = \lambda + o(\lambda)\,,$$

and

$$\Pr(\text{generate} \geq 2 \text{ offspring}) = o(\lambda)\,.$$

In the perspective of a convergence analysis, λ will be thought of as being slowly decreased to zero, like in the simulated annealing algorithm (see [19]). The convergence of annealing schedules will be examined in the next section.

Since our state space V is an arbitrary finite state space, properly defining a mutation operator requires a graph structure, (V, E), that represents how offspring can be generated from the parents. To ensure the irreducibility of the finite Markov chain model for the algorithm, we additionally assume that the graph (V, E) is connected. The mutation operator can then be defined as any particular random walk on the connected graph (V, E).

In the canonical ES, the selection of individuals present in the next generation is usually performed after a deterministic ranking of the parents and offspring. One possible issue with this mode of selection is that random walkers may get trapped into sub-optimal solutions. For example, this can happen if no improvement can be reached by random walking from the last-ranked graph vertex represented in the population. To avoid this issue, we use a slightly modified type of rank-based selection. Instead of selecting μ parents, we actually select $\mu - \Lambda$ parents according to their rank, and then we include the Λ offspring to form the next generation population. This selection scheme requires that the Poisson sampling distribution is conditioned on the event $\Lambda < \mu$, a condition which does not change the above stated key property of the sampling distribution for $\lambda \ll \mu$.

To explain how the modified selection scheme approximates the traditional $(\mu + \lambda)$-ES, we can look at the intermediate generation. After the mutation is applied but before selection is performed, the population consists of the parents plus the offspring,

$$(a_{(1)}, \ldots a_{(\mu)}) + \Lambda \text{ mutant individuals,}$$

where the $a_{(i)}$ denote individuals ranked by decreased order of fitness values. Since we are more specifically interested in the behaviour the algorithm for small values of λ, we can approximate the parental population as μ copies of the current best fit individual, $a_{(1)}$,

$$(a_{(1)}, \ldots, a_{(1)}) + \Lambda \text{ mutant individuals.}$$

Accordingly the loss in diversity when replacing μ parents by $\mu - \Lambda$ parents is generally negligible, if not null. Strongly unfavorable mutations produce offspring viable for one generation, but their carriers usually do not transmit their phenotypes in the subsequent generations. The algorithm behaves similarly to the canonical ES except during peak shifts, which are likely to occur more rapidly in the modified version. Finally, the Poisson-ES for discrete optimisation can be summarised as follows.

The $(\lambda + \mu)$-Poisson-ES. The algorithm iteratively applies the following steps until some stopping criterion is met.

1. Conditional on $\Lambda < \mu$, draw $\Lambda \sim$ Poisson(λ).
2. Generate Λ offspring from the μ parents and apply mutation to the offspring.
3. Select $\mu - \Lambda$ parents according to ranked-based selection.
4. Add the Λ mutant offspring to form the next generation population.

3 Convergence Results for the Poisson-ES

In this section, we describe our main results regarding the hitting time of the optimum for an arbitrary injective objective function f defined on a general search space V when the parameter λ is small. When λ is close to zero, the dynamics of the algorithm are strongly dominated by the selection process, which tends to aggregate individuals into homogeneous (homozygous) populations. Mutations, that usually occur at small rates, can essentially be viewed as perturbations of the selection process [13]. In this context, the behaviour of the algorithm is strongly related to S. Wright's concept of a *fitness landscape* and to the presence of *adaptive valleys* [22]. It has long been acknowledged that the widths and the depths of the adaptive valleys may influence the convergence time of evolutionary algorithms [16]. However, there is a lack of theoretical results that can quantify the convergence rate of an algorithm by means of such geometrical quantities.

Let us represent the fitness landscape by the values of the objective function for each vertex of the graph (V, E). In this section, we define a geometrical quantity that intuitively measures the width of the largest adaptive valley in the fitness landscape. For two vertices a and b, which are viewed as two evolutionary distant individuals by the algorithm, let the distance $d(a, b)$ be defined as the length of the shortest path from a to b in (E, V). In other words, the distance is the minimal number of mutations required to transform a into b. The geometrical quantity of interest is [10,12]

$$\ell_* = \max_{a \neq a_{\text{opt}}} \min_{b : f(b) > f(a)} d(a, b) \, ,$$

where a_{opt} is the (assumed unique) global optimum of f, and a can be restricted to the set of locally sub-optimal phenotypes [10,12]. This quantity measures the greatest distance between a locally optimal individual and a descendent with a higher selective value.

Our main result can be stated as follows.

Theorem 1. *Let (E, V) be a finite connected graph, and let f be an injective function defined on V. Consider the Poisson-ES with parameters $\mu > \lambda$. Let T_{opt} be the hitting time of the optimal solution a_{opt}, and t_{opt} be its expected value*

$$t_{\text{opt}} = \mathrm{E}_v[T_{\text{opt}}] \,,$$

where v is identified to an arbitrary locally optimal population such that $a_{\text{opt}} \notin v$. Then, we have

$$\lim_{\lambda \to 0} \frac{\log(t_{\text{opt}})}{\log(1/\lambda)} = \ell_* \,.$$

In addition, the standard deviation of T_{opt} is equivalent to the expected value

$$\mathrm{sd}[T_{\text{opt}}] \sim t_{\text{opt}} \,, \quad \lambda \to 0 \,.$$

This theorem states that a rough approximation of the mean hitting time can be formulated as

$$t_{\text{opt}} \approx C\lambda^{-\ell_*} \,,$$

for some unknown constant C that depends on μ. Implicitly, the theorem tells us that the spectral gap – that is, one minus the second eigenvalue – of the Markov chain modeling the Poisson-ES is logarithmically equivalent to λ^{ℓ_*} for small λ. Remark that the constant C is not explicit, and may be very large depending on the complexity of the problem under consideration. This happens for example when $\ell_* = 1$, a situation that corresponds to an enumerative sampling strategy. This sort of limitation is also present in the simulated annealing algorithm, where the enumerative strategy leads to a minimum critical depth [14]. In fact, according to [12], and similarly to what has been obtained for the simulated annealing algorithm, the theorem suggests that the convergence towards the optimum can be controlled by a logarithmically decreasing number of offspring

$$\lambda_t = (1 + t)^{-\gamma}, \quad \gamma > \ell_* \,, \quad t \geq 0 \,. \tag{1}$$

To see this, we can introduce an artificial decreasing temperature schedule (T_t) and perform the following change of parameter

$$\lambda_t = e^{-1/T_t} \,.$$

According to [14,12], a necessary and sufficient condition for convergence of the annealed algorithm to the global optimum is then

$$\sum_{t=1}^{\infty} \lambda_t^{\ell_*} = \infty \,,$$

that justifies the form of equation (1) for λ_t.

The proof of the theorem is based on the theory of large deviations applied to Markov chains with rare transitions [13]. It makes use of the Laplace method for computing sums of exponentials. The arguments for the mean hitting time strictly parallel those given in [12] for the Mutation-or-Selection ES (see also [8]). The result for the standard deviation, as well as other results that confirm the geometric-like behaviour of the hitting times, can be derived from [7]. The complete proof is too long to be reproduced here, and we can only give an outline below.

The Mutation-or-Selection ES is based on the following steps. Let p be a mutation probability. At generation t, let $a^t = a$ denote the current population which we also assume to be of fixed size, μ. To update the current state of the population, the algorithm iterates the following operations

1. Select the best individual from the current population, $a_{(1)}$.
2. For each a_i, $i = 1, \ldots, \mu$, either mutate the individual a_i with probability p, or replace it by $a_{(1)}$.

The connection between the MoS-ES and the Poisson-ES arises as the number of offspring in the MoS-ES is also random, and it is distributed according to the Binomial distribution, $\mathrm{Bin}(\mu, p)$. Most of the large deviation analysis is based on the Laplace method as p goes to zero, and makes use of the following key property of the $\mathrm{Bin}(\mu, p)$ distribution

$$\Pr(\text{generate exactly 1 offspring}) = p + o(p)\,,$$

and

$$\Pr(\text{generate} \geq 2 \text{ offspring}) = o(p)\,,$$

which is very similar to the property stated for the Poisson distribution in the previous section. For the MoS-ES [12], we have previously shown that

$$\lim_{p \to 0} \frac{\log(t_{\mathrm{opt}})}{\log(1/p)} = \ell_*\,.$$

In fact, a rough justification of Theorem 1 would consist in setting $\lambda = p/\mu$ and argueing that the $\mathrm{Bin}(\mu, p)$ distribution can be replaced by a Poisson distribution of mean λ according to the classic Binomial-Poisson approximation in probability theory. Although the guess is correct, the argument can easily be seen to be flawed. However the result transfers to the Poisson-ES after a step by step replication of the proof given in [12].

4 Numerical Illustration

To assess the value of the large-deviation approximation for intermediate values of λ, that is,

$$t_{\mathrm{opt}} \approx C\lambda^{-\ell_*}\,,$$

we performed a comparative evaluation of the performances of the Poisson-ES and the MoS-ES on a very simple test problem. The design of experiments and the fit of the simulated data to the large-deviation approximation can be done using an experimental method based on regression, also described in [2,11].

In our example, the objective function was defined on the set of integers $V = [1, \dots, 60]$ as

$$f(x) = 1 + (103 - x)x + 120\sin(x)\,, \quad x \in [1, \dots, 60]\,,$$

and the mutations were implemented as the (reflected) random walk on V. We added ± 1 with equal probability to each a_i in $\{2, \dots, 59\}$. The states 1 and 60 could be moved into 2 and 59, respectively. The corresponding fitness landscape is represented in figure 1. This simple optimisation problem is illustrative of the behaviour of the algorithm for a large class of toy problems. It is easy to predict the behaviour of the algorithm, and to compute some geometrical quantities, and can be generalised to many dimensions without difficulties. We used $\mu = 10$ individuals, and the algorithms were started from the homogeneous population $a^1 = (1, \dots, 1)$. In this example, the adaptive valleys were narrow and easy to cross as we started from a^1, but their width increased as the algorithms moved toward the optimum. The critical parameter ℓ_* was computed as

$$\ell_* = 5\,.$$

In this test problem, the ES were then expected to improve quickly from the starting population, but they were also expected to make slower progress as they approached the global optimum.

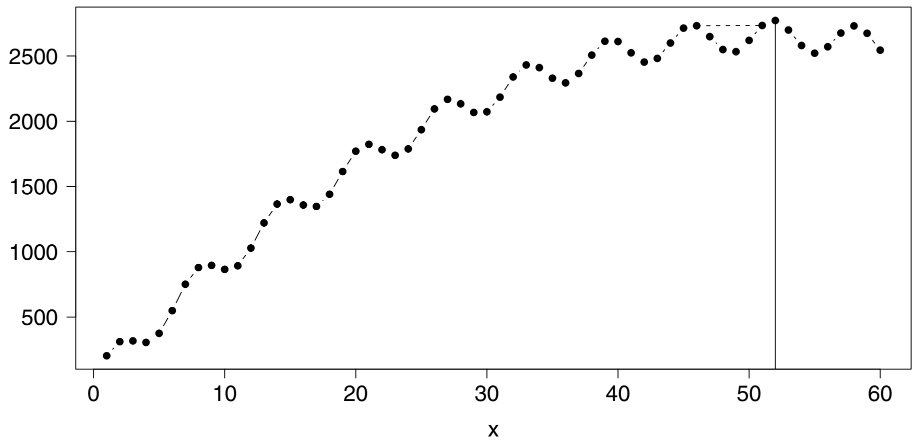


Fig. 1. The toy objective function: $f(x) = 1 + (103 - x)x + 120\sin(x)$, $x = 1, \dots, 60$. The optimum is reached at $x = 52$ and, the width of the largest additive valley is represented by a dashed line. We have $\ell_* = 5$.

To evaluate the performances of the algorithms, we regressed the logarithm of the hitting times on the logarithm of λ (or the logarithm of p) in order to estimate ℓ_* as the slope of the regression

$$\log(T_{\mathrm{opt}}) = \log(C) + \ell_* \log(1/\lambda) + \epsilon \,.$$

To obtain comparable results for the Poisson-ES and for the MoS-ES, we set $p = \lambda/\mu$, and we ran simulations for values of p in the interval $(0.15, 0.4)$, where p is the mutation probability. We obtained 250 replicates of the hitting times for regularly spaced values of p. The corresponding values of λ fall in the interval $(1.5, 4)$. The fact that experimental values of λ were not close to zero makes departures from the theoretical predictions rather likely. These values were nevertheless more conform to standard user-defined ones than would have been the very small values suggested by Theorem 1. Figure 2 shows that the data fit the log-log regression rather well ($R^2 = 0.76$, $P \approx 0$ for the Poisson-ES, and $R^2 = 0.81$, $P \approx 0$ for the MoS-ES), providing evidence that the log-hitting times were actually explained by the logarithm of λ. The coarse approximation $t_{\mathrm{opt}} \approx C\lambda^{-\ell}$ could then considered valid for values of λ not close to zero. The coefficients of the regression model were computed as 3.3 (intercept) and 4.2 (slope) for the Poisson-ES, and they were computed as 2.7 (intercept) and 5.9 (slope) in the MoS-ES. The slope values 4.2 and 5.9 were close to the value $\ell_* = 5$ predicted by the theory of large deviations. In this example, we noticed that the Poisson-ES ran slightly faster than the MoS-ES to the population with the highest selective values.

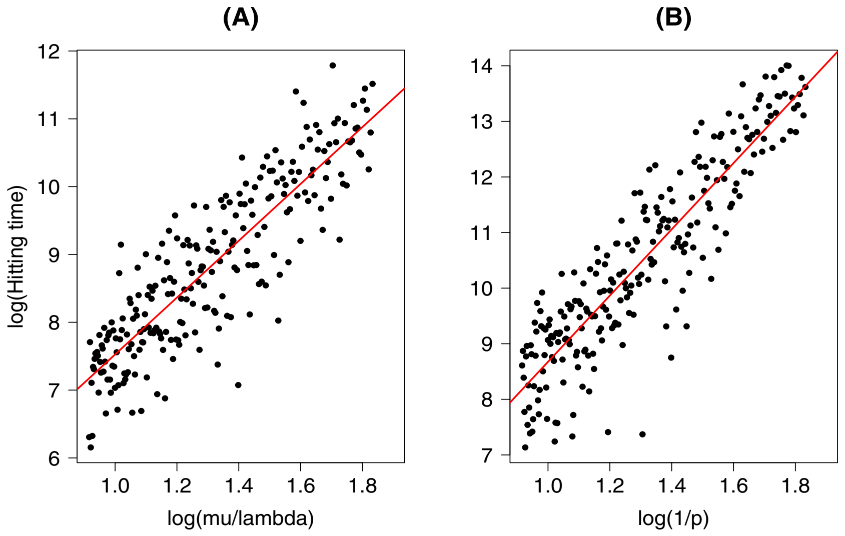


Fig. 2. Regression of the log hitting time on $\log(1/p)$, where p is the mutation probability. The slope of the regression corresponds to the critical quantity ℓ_*. (A) Poisson-ES. (B) MoS-ES, $p = \lambda/\mu$, $\mu = 10$.

5 Discussion

This article has introduced a variant of the canonical ES in which the number of offspring is not deterministic, but is instead sampled from a Poisson distribution with mean λ. After a slight change on the rank-based selection for the μ parents, we showed that the new ES resembles the basic Mutation or Selection-ES introduced in [10]. The Poisson-ES and the MoS-ES provide interesting models for obtaining in-depth insights on the convergence of evolutionary algorithms. Based on the similarity between the two algorithms, we stated a convergence theorem for arbitrary discrete optimization problems, that emphasises the role of the width of the adaptive valleys. Yet, analogs of MoS-ES or of discrete Poisson-ES have not been studied in continuous optimization problems, but it is natural to expect that geometric quantities similar to those influencing the behaviour of the discrete algorithms are likely to determine the convergence rate of the continuous algorithms as well.

Stochastic parameters are the basis for designing adaptive or self-adaptive algorithms. Since the cost of an algorithm is a function of the mutation load through the number of fitness evaluations, an ES should end with $\lambda \approx 0$ when getting close to the optimum. In contrast, being far from the optimum would probably require that the number of offspring is large $\lambda \gg 1$. This idea can be implemented in the Poisson-ES using an explicit convergent annealing scheme. We also believe that this study opens new directions for self-adaptation in discrete ES, because it indicates that increasing λ or performing faster walks in the bottom of the valleys is likely to improve the convergence rate of the algorithm.

Acknowledgments

I wish to thanks Anne Auger for her comments on a previous draft of the manuscript. This work is supported by a grant from the Agence Nationale de la Recherche BLAN06-3146280.

References

1. Auger, A.: Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible Markov chains. Theor. Comput. Sci. 334, 35–69 (2005)
2. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation – The New Experimentalism. Natural Computing Series. Springer, Berlin
3. Beyer, H.-G.: The Theory of Evolution Strategies. Natural Computing Series. Springer, Heidelberg (2001)
4. Beyer, H.-G., Schwefel, H.-P., Wegener, I.: How to analyse evolutionary algorithms. Theor. Comput. Sci. 287, 101–130 (2002)
5. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies – A comprehensive introduction. Natural Computing 1, 3–52 (2002)
6. Bienvenüe, A., François, O.: Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. Theor. Comput. Sci. 306, 269–289 (2003)

7. Cercueil, A., François, O.: Sharp asymptotics for fixation times in stochastic population genetics models at low mutation probabilities. Journal of Statistical Physics 110, 311–332 (2003)
8. Cerf, R.: Asymptotic convergence of genetic algorithms. Adv. Appl. Probab. 30, 521–550 (1998)
9. Droste, S., Jansen, T., Wegener, I.: On the analysis of the $(1 + 1)$ EA. Theor. Comput. Sci. (276), 51–81 (2002)
10. François, O.: An evolutionary algorithm for global minimization and its Markov chain analysis. IEEE Trans. Evol. Comput. 2, 77–90 (1998)
11. François, O., Lavergne, C.: Design of evolutionary algorithms: A statistical perspective. IEEE Trans. Evol. Comput. 5, 129–148 (2001)
12. François, O.: Global optimization with exploration/selection algorithms and simulated annealing. Ann. Appl. Probab. 12, 248–271 (2002)
13. Freidlin, M.I., Wentzell, A.D.: Random Perturbations of Dynamical Systems. Springer, New York (1984)
14. Hajek, B.: Cooling schedules for optimal annealing. Math. Oper. Research 13, 311–329 (1988)
15. Jägersküpper, J.: Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. Theor. Comput. Sci. 379, 329–347 (2007)
16. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1996)
17. Neumann, F., Wegener, I., Randomized, I.: local search, evolutionary algorithms, and the minimum spanning tree problem. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 713–724. Springer, Heidelberg (2004)
18. Rudolph, G.: Finite Markov chain results in evolutionary computation: A tour d'horizon. Fundam. Inform. 35, 67–89 (1998)
19. Schmitt, L.M.: Theory of genetic algorithms. Theor. Comput. Sci. 259, 1–61 (2001)
20. Wegener, I., Witt, C.: On the optimization of monotone polynomials by simple randomized search heuristics. Combin. Probab. Comput. 14, 225–247 (2005)
21. Witt, C.: Runtime Analysis of the $(\mu+1)$ EA on Simple Pseudo-Boolean Functions. Evol. Comput. 14, 65–86 (2006)
22. Wright, S.: The roles of mutation, inbreeding, crossbreeding and selection in evolution. In: Proceedings of the VI International Congress of Genetics, pp. 356–366 (1932)

Multiobjectivization by Decomposition of Scalar Cost Functions

Julia Handl, Simon C. Lovell, and Joshua Knowles

The University of Manchester, UK
{j.handl,simon.lovell,j.knowles}@manchester.ac.uk

Abstract. The term 'multiobjectivization' refers to the casting of a single-objective optimization problem as a multiobjective one, a transformation that can be achieved by the addition of supplementary objectives or by the decomposition of the original objective function. In this paper, we analyze how multiobjectivization *by decomposition* changes the fitness landscape of a given problem and affects search. We find that decomposition has only one possible effect: to introduce plateaus of incomparable solutions. Consequently, multiobjective hillclimbers using no archive 'see' a smaller (or at most equal) number of local optima on a transformed problem compared to hillclimbers on the original problem. When archived multiobjective hillclimbers are considered this effect may partly be reversed. Running time analyses conducted on four example functions demonstrate the (positive and negative) influence that both the multiobjectivization itself, and the use vs. non-use of an archive, can have on the performance of simple hillclimbers. In each case an exponential/polynomial divide is revealed.

1 Introduction

The term 'multiobjectivization' was introduced in [10] to refer to the reformulation of originally single-objective problems as multiobjective ones. Two approaches to this reformulation can be taken, namely the decomposition of the original objective, or the addition of new objectives. In both of these cases, it is a requirement that each of the original optima becomes a Pareto optimum under the new set of objectives [10].

In [10] and in related work, both theoretical and practical [1,7,12,14] it has been demonstrated that such reformulations of problems can in some cases lead to accelerated search performance (comparing broadly equivalent single-objective and multiobjective algorithms). Brockhoff et al. recently presented general theoretical results for adding objectives to a problem, showing that it may have beneficial or detrimental effects on the runtime for a given problem [1]. They show (ibid.) that the addition of objectives to an originally single-objective problem has only two effects on solution orderings: (a) solution pairs that are equal ('indifferent') with regard to the single-objective formulation may become comparable (i.e., one dominates the other), or alternatively, (b) solution pairs that are comparable with regard to the single-objective formulation may become incomparable (i.e., neither dominates the other). Running time analyses (ibid.) comparing a (1+1)-EA and the Global SEMO (Global Simple Evolutionary Multi-objective Optimizer) algorithm indicated that large performance differences (a polynomial speedup and an exponential slowdown) can be observed for a single-objective short path function (derived from SPC_n [6]) when objectives are added.

Here, we are interested in multiobjectivization through decomposition of a single-objective function, which has not been addressed in [1]. After the introduction of some basic notation and algorithms in Section 2, Section 3 goes on to show that the number of possible effects of a decomposition is in fact reduced compared to the scenario discussed in [1], and that this allows for additional inferences regarding the changes to the fitness landscape. It also discusses why these theoretical results only directly apply to algorithms that do not employ an archive. Section 4 of the paper introduces four example problems: the first two illustrate that, despite the theoretical differences to the scenario considered in [1], multiobjectivization through decomposition can equally render a problem easier or harder. The last two are examples of problems where the introduction of an archive makes the problem significantly easier or more difficult. Finally, Section 5 concludes.

2 Notation and Algorithms

Formally, an unconstrained single-objective (scalar) optimization problem is described by the set of feasible solutions X and an objective function $f : X \rightarrow \mathbb{R}$. Without loss of generality, we assume minimization of f, so that the optimal solutions to the problem are those that satisfy

$$\mathrm{argmin}_{x \in X} f(x). \tag{1}$$

Multiobjectivization by decomposition reformulates the problem through the decomposition of the original objective into two or more components. This is achieved through the definition of k objectives $f_i : X \rightarrow \mathbb{R}$, $i \in 1..k$, with the constraint that $f(x) = \sum_{i=1}^{k} f_i(x),\ \forall x$. This transforms the scalar optimization problem into a vector optimization problem with every solution $x \in X$ mapping to a k-tuple of objective values $\boldsymbol{f}(x) = (f_1(x), \ldots, f_k(x))^T$. The optimal solutions to the problem are those that satisfy

$$\mathrm{argmin}_{x \in X} \boldsymbol{f}(x). \tag{2}$$

In vector optimization problems such as (2), a partial ordering of the solutions can be obtained using the concept of Pareto dominance. Solution x is said to dominate solution y, denoted as $\boldsymbol{f}(x) \prec \boldsymbol{f}(y)$, iff $\forall i \in 1..k : f_i(x) \leq f_i(y) \wedge \exists j \in 1..k : f_j(x) < f_j(y)$. The solutions x and y are said to be indifferent, denoted as $\boldsymbol{f}(x) = \boldsymbol{f}(y)$, iff $\forall i \in 1..k : f_i(x) = f_i(y)$. Iff x and y are not indifferent and neither dominates the other, they are said to be incomparable or mutually non-dominated, denoted as $\boldsymbol{f}(x) \sim \boldsymbol{f}(y)$. Under the Pareto framework, the solution to a vector optimization problem is the set of solutions $S \subseteq X$, which are not dominated by any other solutions in the search space: $S = \{s \in X \mid \nexists x \in X : \boldsymbol{f}(x) \prec \boldsymbol{f}(s)\}$. The set of optimal solutions for (1) form a subset of the set of Pareto optimal solutions for formulation (2). Problem (2) can therefore be employed as an alternative formulation to find solutions to (1).

Hillclimbers. The optimizers considered in this paper are very basic single-objective and multiobjective hillclimbers, defined thus:

SOHC. *Initialize a current solution at random. While not done* {*mutate the current solution and accept the mutant iff it is* not worse *than the current* } .
MOHC. *Initialize a current solution at random. While not done* {*mutate the current solution and accept the mutant iff it is* not dominated *by the current* } .
MOHC+A. *Initialize a current solution at random and copy it into the nondominated solutions archive. While not done* {*mutate the current solution and accept the mutant iff* it is not dominated by anything in the archive. *If the mutant is accepted, copy it to the archive. Remove from the archive any solutions that are dominated*} .

The mutation operator used in all three algorithms is $1/n$ bit flip mutation, where n is problem size. The single-objective hillclimber (SOHC) accepts moves to equal cost solutions (cf. [6,2,8]). Equivalently, the basic multiobjective hillclimber (MOHC) rejects the mutant solution only if its objective vector is dominated by the parent solution; it accepts moves to incomparable solutions. For the archiving multiobjective hillclimber (MOHC+A), a mutant solution is accepted only if it is not dominated by either the current solution or a solution in the nondominated solutions archive (similarly to (1+1)-PAES [11]). As discussed in [9] (pages 102–105), this way of using an archive yields a *negative efficiency preserving* strategy [5], i.e., it prevents degradation of solutions. We say there is *degradation* if the current solution is replaced at some later iteration by one that it dominates. Such degradation prevents convergence and can lead to endless cycling between solutions that are not mutually incomparable. N.B., the type of archiving used in the GSEMO algorithm [1] is different because the archive is used as a population from which to select solutions; we do not consider this type of archiving here.

3 Changes to the Fitness Landscape

This section studies the changes in the fitness landscape when moving from a scalar optimization problem to the multiobjective problem obtained through a decomposition of the fitness function as defined in Equation 2.

Definition 1. (Neighborhood) *We define a neighborhood function as a function* $\nu : X \rightarrow X^m$*, with the two properties,* $x \in \nu(x)$*, and* $x \in \nu(y) \leftrightarrow y \in \nu(x)$*. Solutions* x *and* y *are said to be neighbors if* $x \in \nu(y)$ *(and, equivalently,* $y \in \nu(x)$*). The neighborhood size is* m*.*

Definition 2. (Connected sets) *A set of solutions* S *is said to be connected, if* $\forall s, t \in S : \exists p_i \in S : p_1 = s \wedge p_l = t \wedge \forall_{i=1}^{l-1} p_{i+1} \in \nu(p_i)$.

Definition 3. (Local optima) *A set of connected solutions* S *is said to be locally optimal under objective* f *and neighborhood function* ν*, iff* $\forall s \in S : \forall t \in \nu(s) : f(s) < f(t) \vee (t \in S \wedge f(s) = f(t))$*. Equivalently, a set of connected solutions* S *is said to be locally optimal under the set of objectives* $\boldsymbol{f}$ *and neighborhood function* ν*, iff* $\forall s \in S : \forall t \in \nu(s) : \boldsymbol{f}(s) \prec \boldsymbol{f}(t) \vee (t \in S \wedge (\boldsymbol{f}(s) = \boldsymbol{f}(t) \vee \boldsymbol{f}(s) \sim \boldsymbol{f}(t))$*. Every such set* S *is defined as one local optimum.*

Definition 4. (Plateaus) *A set of connected solutions* S *is said to form a plateau under objective* f *and neighborhood function* ν*, iff* $\forall s \in S : \forall t \in S : \exists p_i \in S :$

$p_1 = s \wedge p_l = t \wedge \forall_{i=1}^{l-1}(p_{i+1} \in \nu(p_i) \wedge f(p_{i+1}) = f(p_i))$. Equivalently, a set of connected solutions S is said to form a plateau under objective $\boldsymbol{f}$ and neighborhood function ν, iff $\forall s \in S : \forall t \in S : \exists p_i \in S : p_1 = s \wedge p_l = t \wedge \forall_{i=1}^{l-1}(p_{i+1} \in \nu(p_i) \wedge (\boldsymbol{f}(p_{i+1}) \sim \boldsymbol{f}(p_i) \vee \boldsymbol{f}(p_{i+1}) = \boldsymbol{f}(p_i)))$. A plateau is maximally sized iff $\nexists s \in \{X \setminus S\}$ such that $s \cup S$ is a plateau.

Theorem 1 (Gradient cannot be introduced). *If $f(x) = f(y)$ then $\boldsymbol{f}(x) \sim \boldsymbol{f}(y) \vee \boldsymbol{f}(x) = \boldsymbol{f}(y)$.*

Proof. Assume that $f(x) = f(y)$ and $\boldsymbol{f}(x) \prec \boldsymbol{f}(y)$. Then, by definition of dominance, for each component of $\boldsymbol{f}$, $f_i(x) \leq f_i(y)$ and there exists a component f_j where $f_j(x) < f_j(y)$. As $f(x) = f_1(x) + f_2(x) + \ldots + f_k(x)$, this contradicts the assumption that $f(x) = f(y)$. Thus, if $f(x) = f(y)$, then x does not dominate y. By a symmetric argument, y does not dominate x either. Therefore, by definition of the dominance relations, either x and y are indifferent (the same in all components) or incomparable. □

Theorem 2 (Gradient cannot be reversed). *If $\boldsymbol{f}(x) \prec \boldsymbol{f}(y)$ then $f(x) < f(y)$.*

Proof. Assume that $\boldsymbol{f}(x) \prec \boldsymbol{f}(y)$. Then, by definition of dominance, the objective value of x will be smaller than that of y under f, which, by definition, is the sum of the components of $\boldsymbol{f}$. □

The implication of Theorem 1 and 2 is that decomposition of a scalar cost function has a single consequence for solution orderings: to introduce plateaus.

We next prove that multiobjectivization by decomposition decreases or leaves unchanged the number of local optima in a landscape. We first show that our definition of local optimum ensures that local optima are disjoint sets. This is sufficient to ensure that their number is well-defined. This holds in both the single-objective and the multiobjective spaces. We then show that for each optimum in the multiobjective space there is at least one in the single-objective space. Taken together with the fact that local optima are disjoint, this is sufficient to prove the theorem.

Lemma 1. *The local optima are uniquely defined and are disjoint sets (both in the single-objective and the multiobjective space).*

Proof. The definitions of the local optima above ensure that each one is a maximally sized set and all immediate neighbors of the set are worse. Two local optima cannot contain the same solution because this would necessitate them being in the same optimum; hence the sets are disjoint and uniquely defined. □

Lemma 2. *A locally optimal set in the multiobjective landscape contains at least one locally optimal set in the single-objective landscape.*

Proof. Let S be a set of connected solutions that is locally optimal under $\boldsymbol{f}$. Then, we know that $\forall s \in S : \forall t \in \nu(s) : \boldsymbol{f}(s) \prec \boldsymbol{f}(t) \vee (t \in S \wedge (\boldsymbol{f}(s) = \boldsymbol{f}(t) \vee \boldsymbol{f}(s) \sim \boldsymbol{f}(t))$. If t is not part of S, it follows that s dominates t and, by Theorem 1 and 2, the gradient between s and t remains under the single-objective formulation: $\forall s \in S : \forall t \in \nu(s) : f(s) < f(t) \vee t \in S$. The set S is then split into the following sets: $S^* = \min_{s \in S} f(s)$

and $T^* = S \setminus S^*$. By definition, we know that $\forall s \in S^* : \forall t \in T^* : f(s) < f(t)$. It follows that $\forall s \in S^* : \forall t \in \nu(s) : f(s) < f(t) \vee (t \in S^* \wedge (f(s) = f(t)))$. The set S^* may consist of one or more sets of connected solutions, each of which corresponds to one local optimum (additional optima on different fitness levels are also possible). □

Theorem 3. *The number of local optima under $\boldsymbol{f}$ is smaller or equal to the number of local optima under f.*

Proof. The result follows directly from the two previous lemmata. □

How does this affect search? An increase in the number and size of plateaus may contribute to making a problem more difficult, as, on such plateaus, optimization methods have to succumb to random walk behavior. On the other hand, a reduction in the number of local optima may mitigate the difficulty of search. Both effects are intimately linked, as the removal of local optima is only possible through the creation of a plateau: it requires the creation of at least one path of incomparable solutions that crosses the fitness barrier around the original local optimum. Whether a problem gets easier or more difficult as a result of the decomposition will thus depend on the number and size of the plateaus that are created and the parts of the fitness landscape they replace. In Section 4.1. and 4.2., we introduce two examples of decompositions that cause an exponential increase or decrease in the runtime required by a multiobjective hillclimber compared to the runtime required by a single-objective hillclimber on the corresponding single-objective problem.

What about archives? The introduction of an archive (of the type used in MOHC+A) has the effect of restricting movement along plateaus of incomparable solutions, dependent upon what solutions are in the archive. On one hand, this means that, from the point of view of the algorithm, some of the local optima removed by the decomposition may again be perceived. On the other hand, degrading moves are prevented, which may have a beneficial effect on search. In Section 4.3. and 4.4., we introduce functions that are examples of problems where the introduction of an archive makes a multiobjective function (alternately) harder or easier.

4 Four Example Functions

In this section, we consider functions $\boldsymbol{f} : \{0,1\}^n \rightarrow \mathbb{R}^2$ (with restrictions on n for some problems), each being a decomposition of a different single objective function f. In our discussion of these functions, we consider local optima as defined by the neighborhood function $\nu(x) = \{y | H(x,y) \leq 1\}$, where $H(x,y)$ is the Hamming distance of two bit strings defined as $H(x,y) = \sum_{i=1}^{n} |x_i - y_i|$. We provide proof sketches[1] regarding the derivation of theoretical bounds on the runtime required by the different algorithms to reach the global optimum and give empirical results on the number of evaluations taken (means and standard errors over 20 runs; their approximate fitting with analytical curves agreeing with the theoretical bounds on time complexity is also shown).

[1] Proof details can be obtained from the first author.

4.1 Decomposition Can Make a Problem Harder

We define the function $\mathrm{slope}(x) = (f_1(x), f_2(x))^T$ over the set of binary strings $x \in \{0,1\}^n$:

$$f_1(x) = |x|_1 \qquad\qquad f_2(x) = 2n - 2|x|_1,$$

where $|x|_1$ gives the number of ones in binary string x. The effect of multiobjectivization in this example is the generation of a plateau of exponential size. A plot of the function and empirical results are given in Figure 1.

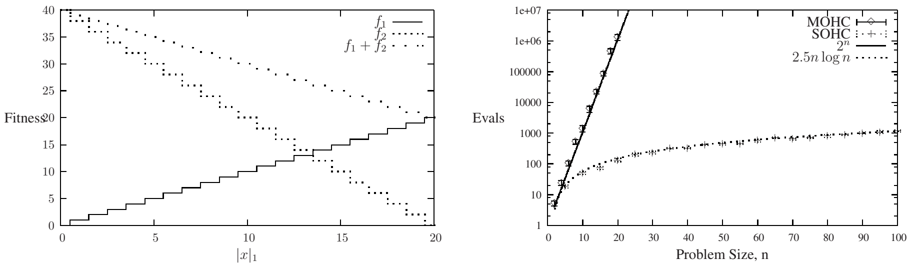


Fig. 1. Function $\mathrm{slope}(x)$. Left: Function for $n = 20$. Right: Empirical results.

We consider the runtime required by SOHC and MOHC to reach the global optimum of $f_1(x)+f_2(x)$ at $|x|_1 = n$. SOHC reaches this optimum by hillclimbing (descending) the monotonic slope towards it. The expected runtime for the problem is equivalent to that of MAXONES, which is $\Theta(n \log n)$ [6]. For MOHC every solution is incomparable with respect to every other. Therefore, the expected waiting time for MOHC is equivalent to that of the (1+1)-EA on the needle-in-a-haystack function, and is given as $\Theta(2^n)$ [3]. MOHC+A performs identically to MOHC, as degradation of solutions cannot occur for this problem.

4.2 Decomposition Can Make a Problem Easier

The function we use here is inspired by the short path function defined in [6], which was used as a basis for some other functions in [1].

We define the function $\mathrm{shortpath}(x) = (f_1(x), f_2(x))^T$ over the set of binary strings $x \in \{0,1\}^n$:

$$f_1 = \begin{cases} 2|x|_1 & \text{if } x \in \{1^i0^{n-i}, i \in 1..n\} \wedge |x|_1 \leq n-1 \\ 0 & \text{if } x \in \{1^i0^{n-i}, i \in 1..n\} \wedge |x|_1 = n \\ 2n + |x|_1 & \text{otherwise,} \end{cases}$$

$$f_2 = \begin{cases} n - |x|_1 & \text{if } x \in \{1^i0^{n-i}, i \in 1..n\} \\ 2n + |x|_1 & \text{otherwise.} \end{cases}$$

The effect of multiobjectivization is the introduction of a plateau that is a short path and the removal of the local optimum at $|x|_1 = 0$. A plot of this crucial part of the function and empirical results are given in Figure 2.

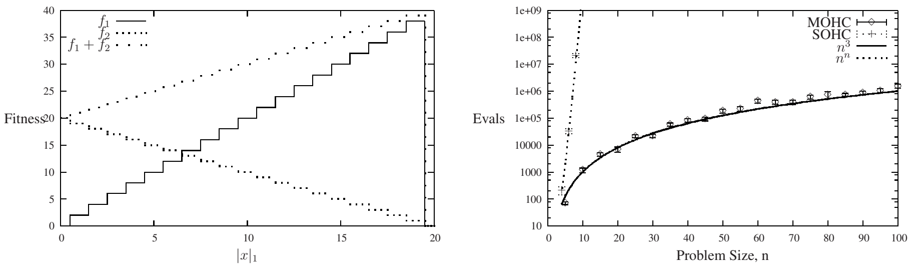


Fig. 2. Function $\mathrm{shortpath}(x)$. Left: Part of the function for ($x \in \{1^i 0^{n-i}, i \in 1..n\}$) for $n = 20$. Right: Empirical results.

We consider the runtime required by SOHC and MOHC to reach the global optimum of $f_1(x) + f_2(x)$ at $|x|_1 = n$. MOHC behaves equivalently to the (1+1)-EA on function $\mathrm{SPC_n}$ [6] and its expected runtime is therefore bounded by $O(n^3)$. MOHC+A performs identically to MOHC, as degradation of solutions cannot occur for this problem.[2] Regarding the expected runtime of SOHC, a lower bound of $n^{\Omega(n)}$ can be shown similarly to the analysis of the (1+1)-EA on $\mathrm{SPC_n}$ in [6].

4.3 An Archive Can Make a Problem Harder

We define the function $\mathrm{barrier}(x) = (f_1(x), f_2(x))^T$ over the set of binary strings $x \in \{0,1\}^n$ for $n \bmod 10 = 0$:

$$f_1(x) = \begin{cases} n - |x|_1 & \text{if} |x|_1 \leq 0.9n - 2 \\ 1.1n - |x|_1 + 2 & \text{otherwise,} \end{cases}$$

$$f_2(x) = \begin{cases} n - |x|_1 & \text{if} |x|_1 \leq 0.9n - 1 \\ 1.1n - |x|_1 + 1 & \text{otherwise.} \end{cases}$$

The effect of the use of an archive is to make parts of a plateau inaccessible. A plot of the function and empirical results are given in Figure 3.

We consider the runtime required by MOHC and MOHC+A to reach the global optimum of $f_1(x) + f_2(x)$ at $|x|_1 = n$. Function $\mathrm{barrier}(x)$ has a plateau of incomparable solutions at $0.9n - 2 \leq |x|_1 \leq 0.9n$. With probability exponentially close to 1, the hillclimbers are initialized with a solution with $|x|_1 < 0.9n - 2$ (this follows by Chernoff's bounds, see [4,6]), and will need to cross the plateau to reach $|x|_1 = n$. MOHC will reach the plateau in expected time $\Theta(n \log n)$. It will then perform a random walk on the plateau until it succeeds in adding at least three more ones to the bit string. A lower bound on the probability of this success can be obtained as the probability of performing three drift moves to the right directly in sequence, which is given as $\prod_{k=0.1n}^{0.1n+2} \frac{k}{n} (\frac{n-1}{n})^{(n-1)}$. The waiting time for this event is $O(1)$. Once the plateau has

[2] The possibility of degradation can be readily introduced into a function of the same structure through the adjustment of relative fitness levels between solutions that lie within and outside of $x \in \{1^i 0^{n-i}, i \in 1..n\}$, so that MOHC can fall off the short path. In that case, MOHC+A will outperform MOHC.

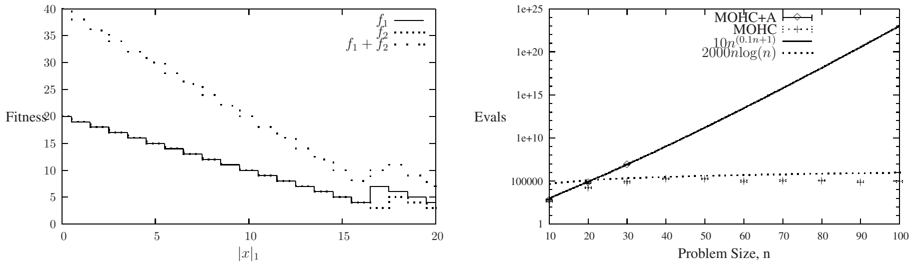


Fig. 3. Function barrier(x). Left: Function for $n = 20$. Right: Empirical results.

been crossed, any mutation that increases $|x|_1$ will be accepted, and the probability of such a mutation is at least $\Omega\left(\frac{n-|x|_1}{n}\right)$. If only such mutations were possible, the expected waiting time for reaching the global optimum would be $O(n\log n)$. However, mutations reducing $|x|_1$ to $0.9n - 2 \leq |x|_1 \leq 0.9n - 1$ can also be accepted: these require at least $k = |x|_1 - 0.9n + 1$ simultaneous bit flips and have probability at most $O\left(\frac{1}{k!}\right)$. At position $|x|_1$, a lower bound on the probability of an increase in $|x|_1$ to happen before a decrease in $|x|_1$ is then given as $p_k = \frac{k!}{k! + \frac{n}{0.1n-k+1}}$. Multiplying over all $0.9n + 1 \leq |x|_1 \leq n - 1$ yields $\prod_{k=2}^{0.1n} p_k$, which converges. Therefore, the possible returns to the plateau only increase the expected waiting time by a constant factor, and the overall expected runtime of MOHC is $O(n\log n)$.

MOHC+A will reach the plateau in expected time $\Theta(n\log n)$. The archive then prevents the access of all the solutions with $0.9n \leq |x|_1 \leq n - 1$. In order to reach the global optimum at $|x|_1 = n$, it will need to set all remaining $0.1n + 1$ or $0.1n + 2$ bits with value zero to one simultaneously. The probability of such a mutation is bounded above by $O((\frac{1}{n})^{(0.1n+1)})$. The lower bound on the overall runtime of the MOHC+A is therefore given as $n^{\Omega(n)}$.

4.4 An Archive Can Make a Problem Easier

We define the function $\mathrm{steps}(x) = (f_1(x), f_2(x))^T$ over the set of binary strings of even size: $x \in \{0,1\}^n$ for $n \bmod 2 = 0$.

$$f_1(x) = \begin{cases} 0.5n - 0.5|x|_1 & \text{if} |x|_1 \bmod 2 = 0 \\ 0.5n - 0.5|x|_1 + 1 & \text{otherwise,} \end{cases}$$

$$f_2(x) = \begin{cases} 0.5n - 0.5|x|_1 & \text{if} |x|_1 \bmod 2 = 0 \\ 0.5n - 0.5|x|_1 - 2 & \text{otherwise.} \end{cases}$$

The effect of the use of an archive is to prevent degradation on this plateau of neighboring non-dominated solutions. A plot of the function and empirical results are given in Figure 4.

We consider the runtime required by MOHC and MOHC+A to reach the global optimum of $f_1(x) + f_2(x)$ at $|x|_1 = n$. In order to reach $|x|_1 = n$, MOHC+A needs to increase the number of ones and drift through the plateaus that separate it from the

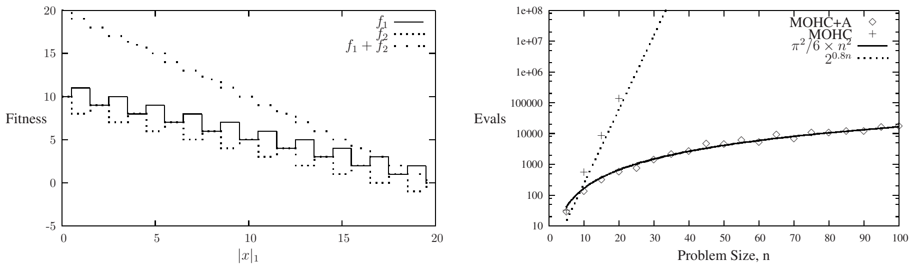


Fig. 4. Left: Function $\mathrm{steps}(x)$ for $n = 20$. Right: Empirical results.

global optimum. Alternatively, it can overcome each such plateau by two simultaneous mutations and, at position $|x|_1$, the probability of increasing the number of ones by two is therefore bounded below by $\binom{n-|x|_1}{2}(\frac{1}{n})^2(\frac{n-1}{n})^{n-2}$. Summing over all values of $|x|_1$ then yields an upper bound of $O(n^2)$ on the expected runtime of MOHC+A (as $\sum_{|x|_1=0}^{n} \frac{1}{|x|_1^2}$ converges).

In contrast to this, MOHC may drift back to $|x|_1 = 0$ from any point in the search space (using single bit flip mutations). In order to show exponential runtime we make use of the drift theorem introduced by Oliveto and Witt [13]. Let $\triangle(i)$ denote the random increase in the number of zeros when mutating a bit string with $|x|_1 = i$. We now need to identify an interval $[a, b]$ of asymptotic size on $0 \leq |x|_1 \leq n$ for which it can be shown that (1) $P(\triangle(i) = -j) \leq \frac{1}{1+\delta}^{j-r}$ for $i > a$ and $j \geq 1$ and (2) $E(\triangle(i)) > \epsilon$, for $a < i < b$, where δ, r and ϵ are constants. As shown in [13], Condition 1 holds for the (1+1)-EA independently of i and of acceptance, which also carries over to our analysis. For $a = \frac{2}{3}n$ and $b = n$, it can be shown that $E(\triangle(i)) > \frac{1}{6}e^{-1}$, for $a < i < b$, which fulfills Condition 2. It follows that the probability of finding the global optimum in 2^{cn} steps is at most $2^{-\Omega(n)}$.

5 Conclusion

This paper has considered the transformation of scalar optimization problems into multiobjective ones. Where the multiobjective problem is obtained through a decomposition of the original objective function, this can cause only one type of change to solution orderings: pairs of solutions that initially had different scalar objective values can become incomparable. Running time analyses and empirical results for both archiving and non-archiving algorithms show that, dependent on the function, a problem can become easier or harder. This can be understood in terms of the introduction and removal of plateaus and local optima.

Acknowledgments

We thank the anonymous reviewers for their many helpful comments and advice. JH gratefully acknowledges support by a Special Training Fellowship from the Medical

Research Council (MRC), UK. JK is supported by a David Phillips Fellowship from the Biotechnology and Biological Sciences Research Council (BBSRC), UK.

References

1. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do additional objectives make a problem harder. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 765–772. ACM Press, New York (2007)
2. Forrest, S., Mitchell, M., Whitley, L.: Relative Building-Block Fitness and the Building-Block Hypothesis. In: Foundations of Genetic Algorithms 2, pp. 109–126. Morgan Kaufmann, San Mateo (1993)
3. Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. Evolutionary Computation 7(2), 173–203 (1999)
4. Hagerub, T., Rüb, C.: A guided tour of Chernoff bounds. Information Processing Letters 33, 305–308 (1989)
5. Hanne, T.: On the convergence of multiobjective evolutionary algorithms. European Journal of Operational Research 117(3), 553–564 (1999)
6. Jansen, T., Wegener, I.: Evolutionary algorithms — how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Transactions on Evolutionary Computation 5(6), 589–599 (2001)
7. Jensen, M.T.: Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation. Journal of Mathematical Modelling and Algorithms 3(4), 323–347 (2004)
8. Juels, A., Wattenberg, M.: Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems 8, pp. 430–436. MIT Press, Cambridge (1995)
9. Knowles, J.: Local-search and hybrid evolutionary algorithms for Pareto optimization. PhD thesis, University of Reading, UK (2002)
10. Knowles, J., Watson, R., Corne, D.: Reducing local optima in single-objective problems by multi-objectivization. In: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, pp. 269–283. Springer, Berlin (2001)
11. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)
12. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Natural Computing 5(3), 305–319 (2006)
13. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. In: Rudolph, G., et al. (eds.) PPSN X 2008. LNCS, vol. 5199, pp. 82–91. Springer, Berlin (2008)
14. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. Journal of Mathematical Modelling and Algorithms 3(4), 346–366 (2004)

A Blend of Markov-Chain and Drift Analysis

Jens Jägersküpper[*]

Technische Universität Dortmund, Informatik 2, 44221 Dortmund, Germany

Abstract. In their seminal article [Theo. Comp. Sci. 276(2002):51–82] Droste, Jansen, and Wegener present the first theoretical analysis of the expected runtime of a basic direct-search heuristic with a global search operator, namely the (1+1) Evolutionary Algorithm ((1+1) EA), for the class of linear functions over the search space $\{0,1\}^n$. In a rather long and involved proof they show that, for any linear function, the expected runtime of the (1+1) EA is $O(n \log n)$, i. e., that there are two constants c and n' such that, for $n \geq n'$, the expected number of iterations until a global optimum is generated is bound above by $c \cdot n \log n$. However, neither c nor n' are specified – they would be pretty large. Here we reconsider this optimization scenario to demonstrate the potential of an analytical method that makes use not only of the drift (w. r. t. a potential function, here the number of bits set correctly), but also of the distribution of the evolving candidate solution over the search space $\{0,1\}^n$: An invariance property of this distribution is proved, which is then used to derive a significantly better lower bound on the drift. Finally, this better estimate of the drift results in an upper bound on the expected number of iterations of $3.8\, n \log_2 n + 7.6 \log_2 n$ for $n \geq 2$.

1 Introduction

We consider the optimization of a pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$ which is given by a black box. That is, knowledge about f can solely be gathered by evaluating f at a number of search points. Black-box optimization is also commonly referred to as *direct search.* Randomized neighborhood search is a commonly used heuristic for direct search in $\{0,1\}^n$. It is an iterative method which tries (in each iteration) to pick a better solution from the neighborhood of the current candidate solution by choosing one of the neighbors uniformly at random. Usually, the neighborhood of $x \in \{0,1\}^n$ consists of the Hamming neighbors of x, i. e. of all $y \in \{0,1\}^n$ with a Hamming distance $H(x,y) = 1$ (number of bits that differ). This heuristic is often called randomized local search (RLS). Obviously, RLS cannot escape a local optimum x^* that is not globally optimal and for which all neighbors have a worse function value. Naturally, a different neighborhood could be chosen. As nothing is known about f, however, this is hard choice to make. A different approach is to consider the complete search space as the neighborhood, but to not sample uniformly at random any longer. One of these so-called global methods is the (1+1) EA. It starts with a search point (called *individual*) which is uniformly chosen from $\{0,1\}^n$. Then, in each iteration, a new candidate solution is generated by *mutation,* namely by independently flipping each bit of the current

[*] Supported by the German Research Foundation (DFG) through the collaborative research center, Computational Intelligence" (SFB 531).

candidate solution with a predefined probability p. Usually, $p := 1/n$ is chosen, and this *mutation rate* will also be considered here in the following. Iff the f-value of the mutant is at least as good, then the mutant becomes the next iterate (otherwise the mutant is discarded, so that the search stays where it was), which is called *elitist selection.*

Note that global convergence is trivially proved for the (1+1) EA: Since in each iteration a global optimum is sampled with probability at least p^n, the expected number of steps until a global optimum is generated is bounded above by p^{-n}, namely by n^n for $p = 1/n$. So (global) convergence is not the point. The point is: How long does it actually take? That is: What is the expected number of steps until a global optimum is generated? It turned out that, because of its global search operator, the (1+1) EA is often much harder to analyze than RLS. After first analyses (for simple functions like OneMax) using Markov chain theory, cf. [1], a different analytical approach from the field of classical algorithmics enabled a bunch of results for less trivial function scenarios: the potential method. Instead of considering the f-value (of the evolving individual), a potential function is defined (w. r. t. the process), which takes its maximum value if and only if the f-value is best. In many such analyses of the (1+1) EA, the number of bits set correctly is considered as the potential (of a search point). This technique was used by Droste, Jansen, and Wegener [2] to prove, among other results, that the expected number of iterations the (1+1) EA needs to generate the optimum of a linear function is $O(n \log n)$. The expected change in the potential (per iteration) is commonly called *drift.* Drift analysis has been put forward by He and Yao [3], for instance. Unfortunately, the application of their quite general approach to the scenario considered here, namely the (1+1) EA using the standard mutation rate $p = 1/n$ to maximize a linear function, contains a flaw [4], so that this fundamental scenario is not covered. Here we reconsider this scenario and show how to prove a significantly better lower bound on the potential's drift using an invariance property of the distribution of the evolving individual over the search space $\{0,1\}^n$ during the optimization process.

The scenario. We consider the class of linear functions over $\{0,1\}^n$ consisting of all $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) := c + \sum_{i=1}^{n} c_i \cdot x_i$, where $x = x_n \cdots x_1$. We assume that f depends essentially on all n bits, i. e., we assume $c_i \in \mathbb{R} \setminus \{0\}$. The coefficient c_i will also be called *the weight* of the ith bit. Solely for better legibility, we assume $c_n \geq c_{n-1} \geq \cdots \geq c_1 > c = 0$. Obviously, the ordering of the bits is irrelevant. Moreover, because of the uniformly random initialization and the invariance property of the mutation operator of the (1+1) EA (it does not care about whether a bit is 1 or 0), these assumptions can be made without a loss of generality: The (1+1) EA behaves identically on f and $f_{\oplus y}(x) := f(x \oplus y)$ for any fixed $y \in \{0,1\}^n$, in particular for y the complement of the optimum ("$\oplus$" denotes the bitwise XOR operation).

Two linear functions which are frequently considered are OneMax, where $c_i := 1$ for each weight, and BinValue, where the bit-string is taken as the binary representation, i. e., $c_i := 2^{i-1}$. These are two extremes in the class of linear functions: In OneMax all bits have the same weight, whereas in BinValue the weight of a single bit (namely the nth) is larger than the total weight of the other $n-1$ bits.

Unless stated differently, we consider the maximization of a linear function with positive weights (non-decreasing from left to right; as described above) by the (1+1) EA (as described above) using the standard mutation rate (bit-flip probability) $p := 1/n$.

2 Invariance Property of the Individual's Distribution over $\{0,1\}^n$

In this section a particular – actually intuitive – property of the distribution of the evolving individual x over the search space $\{0,1\}^n$ will be proved. This will be used in the following Section 3 to prove that *bad mutations,* namely mutations that would result in a loss of 1-bits if they were accepted, are likely to be such that they are discarded by elitist selection. In Section 4 this observation will enable us to prove a significantly better lower bound on the drift (expected change of the number of 1-bits), which is then used to obtain our main result, a better upper bound on the expected optimization time.

Now, consider two bits x_i and x_j in x with $j > i$ (not necessarily adjacent), so that $c_j \geq c_i$ for their weights. First consider the case $x_j x_i = 00$. Assume that a mutation flips x_i, but not x_j, and maybe some more bits. If this mutation is accepted, then the mutation that flips the same bits except for x_j instead of x_i would also be accepted. Note that the latter mutation occurs with the same probability. Now consider the case $x_j x_i = 11$ and assume that a mutation flips x_j, but not x_i, and maybe some more bits. If this mutation is accepted, then the mutation that flips the same bits except for x_i instead of x_j would also be accepted. And again, the latter mutation occurs with exactly the same probability. Thus, since $c_j \geq c_i$, for x_j a change from 0 to 1 is at least as probable as for x_i, whereas a change from 1 to 0 is at most as probable. All in, observing $x_j = 1$ seems at least as probable as observing $x_i = 1$ during the optimization – whatever the number of iterations. This can be formulated more formally:

Theorem 1. *Let $x^{[t]}$ denote the random individual (distributed over $\{0,1\}^n$) after t iterations in our scenario. Then* $\mathrm{Prob}(x_n^{[t]} = 1) \geq \ldots \geq \mathrm{Prob}(x_1^{[t]} = 1)$ *for* $t \geq 0$.

This invariance property of the distribution of the evolving individual over $\{0,1\}^n$ will be proved in the remainder of this section. The superscript in "$x^{[t]}$" will be dropped (unless necessary). Note that „$\mathrm{Prob}(x_j = 1) \geq \mathrm{Prob}(x_i = 1)$" is equivalent to „$\mathrm{Prob}(x_j x_i = 10) \geq \mathrm{Prob}(x_j x_i = 01)$" since

$$\begin{aligned}\mathrm{Prob}(x_j = 1) &= \mathrm{Prob}(x_j x_i = 10) + \mathrm{Prob}(x_j x_i = 11)\\ \mathrm{Prob}(x_i = 1) &= \mathrm{Prob}(x_j x_i = 01) + \mathrm{Prob}(x_j x_i = 11)\end{aligned}$$

Thus, to prove our conjecture we merely have to show that for any pair of adjacent bits in $x \in \{0,1\}^n$, i. e. for $i \in \{0,\ldots,n-2\}$

$$\begin{aligned}&\sum_{X\in\{0,1\}^i, Y\in\{0,1\}^{n-2-i}} \mathrm{Prob}(x = X10Y) \geq \sum_{X\in\{0,1\}^i, Y\in\{0,1\}^{n-2-i}} \mathrm{Prob}(x = X01Y)\\ \iff &\sum_{X\in\{0,1\}^i, Y\in\{0,1\}^{n-2-i}} \big(\mathrm{Prob}(x = X10Y) - \mathrm{Prob}(x = X01Y)\big) \geq 0 \qquad (1)\end{aligned}$$

"$XY \in \{0,1\}^j$" abbreviates "$X, Y \in \{0,1\}^* \wedge |X| + |Y| = j$" in the following. Then

$$(\forall XY \in \{0,1\}^{n-2})\ \mathrm{Prob}(x = X10Y) - \mathrm{Prob}(x = X01Y) \geq 0$$

is sufficient for each of the $n-1$ sums in Eqn. (1) to be non-negative. Thus, our intermediate objective is to prove that for the evolving individual x

$$(\forall XY \in \{0,1\}^{n-2})\ \operatorname{Prob}(x^{[t]} = X10Y) \geq \operatorname{Prob}(x^{[t]} = X01Y) \tag{2}$$

throughout the complete optimization process, i. e. for $t \geq 0$. We will use induction on the number t of iterations to prove Eqn. (2). For the induction step, recall that the optimization of the (1+1) EA is a Markov chain with state space $\{0,1\}^n$. The transition probabilities obviously depend on the function f to be maximized. Let $H(\cdot,\cdot)$ denote the Hamming distance between two bit-strings and let $p(h) = p^h(1-p)^{n-h}$ denote the probability that a mutation flips exactly h bits at particlar positions in the string. Then after iteration $t \geq 1$ the Markov chain is in state $y \in \{0,1\}^n$ with probability

$$\begin{aligned}\operatorname{Prob}(x^{[t]} = y) = \operatorname{Prob}(x^{[t-1]} = y) \cdot \sum_{w \in \{0,1\}^n:\ f(w)<f(y)} p(H(y,w)) \\ + \sum_{z \in \{0,1\}^n:\ f(z)\leq f(y)} \operatorname{Prob}(x^{[t-1]} = z) \cdot p(H(z,y))\end{aligned} \tag{3}$$

The first sum, weighted by $\operatorname{Prob}(x^{[t-1]} = y)$, equals the probability to generate a worse mutant from y, whereas the second sum equals the probability that the mutant of the current state z in the ith step is y, where z can be any state not better than y. This identity will be used in the induction step of the proof of Eqn. (2). Before we do so, however, we take a closer look at the probability to generate a worse mutant.

Lemma 1. *Let* $\operatorname{Mut} : \{0,1\}^n \to \{0,1\}^n$ *denote the random mapping induced by the mutation operator. Then in our scenario for all* $XYZ \in \{0,1\}^{n-2}$*:*
$\operatorname{Prob}\big(f(\operatorname{Mut}(X1Y0Z)) < f(X1Y0Z)\big) \geq \operatorname{Prob}\big(f(\operatorname{Mut}(X0Y1Z)) < f(X0Y1Z)\big)$.

This is almost obvious since $X1Y0Z$ and $X0Y1Z$ have the same number of 1-bits and $f(X1Y0Y) \geq f(X0Y1Z)$. A formal proof can be found in [5]. This result is now used to estimate the transition probabilities in Eqn. (3) within the proof of the following lemma (which implies Theorem 1, the invariance property).

Lemma 2. *In our scenario, after any number of iterations, i. e. for* $t \geq 0$*, the distribution of the evolving individual* $x^{[t]}$ *over* $\{0,1\}^n$ *is such that for all* $XYZ \in \{0,1\}^{n-2}$
$\operatorname{Prob}(x^{[t]} = X1Y0Z) \geq \operatorname{Prob}(x^{[t]} = X0Y1Z)$.

Proof. The induction basis is trivial: Since $x^{[0]}$ is uniformly distributed over $\{0,1\}^n$, $X1Y0Z$ and $X0Y1Z$ are equiprobable after initialization. For the induction step let $XYZ \in \{0,1\}^{n-2}$ be arbitrary, but fixed. Now consider Eqn. (3) for $y := X0Y1Z$ and for $y := X1Y0Z$, telling us the probabilities of the Markov chain being in state $X1Y0Z$ resp. in state $X0Y1Z$ after t iterations (where $p(h) = p^h(1-p)^{n-h}$ is the probability that a mutation flips exactly h particular bits, where $p \in (0,1/2)$ is the bit-flip probability of the mutation operator). For the induction step we need to show $\operatorname{Prob}(x^{[t]} = X1Y0Z) \geq \operatorname{Prob}(x^{[t]} = X0Y1Z)$. Lemma 1 actually tells us

$$\sum_{u \in \{0,1\}^n:\ f(u)<f(X1Y0Z)} p\big(H(X1Y0Z,u)\big) \geq \sum_{w \in \{0,1\}^n:\ f(w)<f(X0Y1Z)} p\big(H(X0Y1Z,w)\big),$$

the induction hypothesis $\text{Prob}(x^{[t-1]} = X1Y0Z) \geq \text{Prob}(x^{[t-1]} = X0Y1Z)$, so that

$$\begin{aligned}
&\text{Prob}(x^{[t-1]} = X1Y0Z) \cdot \sum_{u\in\{0,1\}^n:\ f(X1Y0Z)>f(u)} p\bigl(H(X1Y0Z, u)\bigr) \\
\geq\ &\text{Prob}(x^{[t-1]} = X0Y1Z) \cdot \sum_{w\in\{0,1\}^n:\ f(X0Y1Z)>f(w)} p\bigl(H(X0Y1Z, w)\bigr).
\end{aligned}$$

In other words, the probability of being in state $X1Y0Z$ and staying there (during the tth iteration) is at least as large as it is for $X0Y1Z$. It remains to be shown that getting into state $X1Y0Z$ in the tth iteration is at at least as probable as getting into $X0Y1Z$ in that step. Therefore note that flipping a set of i particular bits is more probable than flipping a set of $i+2$ particular bits when using a bit-flip probability of $p \in (0, 1/2)$ because $0 < p < 1/2 \implies (1-p)^{j-i} > p^{j-i} \iff p^i(1-p)^{n-i} > p^j(1-p)^{n-j}$ for $0 \leq i < j \leq n$. We focus on the rest of the summands, namely we are going to show the following sufficient inequality:

$$\begin{aligned}
&\sum_{u\in\{0,1\}^n:\ f(u)\leq f(X1Y0Z)} \text{Prob}(x^{[t-1]} = u) \cdot p\bigl(H(u, X1Y0Z)\bigr) &&=: S_{10} \\
\geq &\sum_{w\in\{0,1\}^n:\ f(w)\leq f(X0Y1Z)} \text{Prob}(x^{[t-1]} = w) \cdot p\bigl(H(w, X0Y1Z)\bigr) &&=: S_{01}
\end{aligned}$$

Note that any index w in S_{01} occurs also as u in S_{10} since $f(w) \leq f(X0Y1Z) \leq f(X1Y0Z)$. In the following $A \in \{0,1\}^{|X|}$, $B \in \{0,1\}^{|Y|}$, $C \in \{0,1\}^{|Z|}$ and $h := H(ABC, XYZ)$. We consider different cases for the summation index w in S_{01}:

<u>$w = A0B0C$:</u> Since $ABC \in \{0,1\}^{n-2}$ such that $f(A0B0C) \leq f(X0Y1Z)$, and since $f(X0Y1Z) \leq f(X1Y0Z)$, also $f(A0B0C) \leq f(X1Y0Z)$, so that $u = A0B0C$ is an index in S_{10}, too. Finally, the Hamming distance of $A0B0C$ from $X0Y1Z$ as well as from $X1Y0Z$ equals $h+1$, respectively, so that the summand associated with the index $A0B0C$ in S_{10} equals the summand associated with $A0B0C$ in S_{10}.

<u>$w = A1B1C$:</u> This case is analogous to the case $w = A0B0C$ above.

<u>$w = A1B0C$:</u> This implies that $w = A0B1C$ is also an index in S_{01}. Furthermore, recall that these two indices necessarily occur in S_{10}, too. Thus, in this case

$$\begin{aligned}
&\text{Prob}(x^{[t]} = X0Y1Z \mid x^{[t-1]} \in \{A0B1C, A1B0C\}) \\
&\qquad\qquad \leq \text{Prob}(x^{[t]} = X1Y0Z \mid x^{[t-1]} \in \{A0B1C, A1B0C\})
\end{aligned}$$

$$\Leftrightarrow \quad \left.\begin{aligned}&\text{Prob}(x^{[t-1]} = A0B1C) \cdot p(h)\ + \\ &\text{Prob}(x^{[t-1]} = A1B0C) \cdot p(h+2)\end{aligned}\right\} \leq \left\{\begin{aligned}&\text{Prob}(x^{[t-1]} = A0B1C) \cdot p(h+2)\ + \\ &\text{Prob}(x^{[t-1]} = A1B0C) \cdot p(h)\end{aligned}\right.$$

$$\Leftrightarrow \text{Prob}(x^{[t-1]}{=}A0B1C) \cdot (p(h) - p(h{+}2)) \leq \text{Prob}(x^{[t-1]}{=}A1B0C) \cdot (p(h) - p(h{+}2))$$

where the last inequality holds because of the induction hypothesis and $p(h) > p(h{+}2)$, i. e., $p(h) - p(h+2) > 0$ as seen above. In other words, the two summands associated with the indices $A0B1C$ and $A1B0C$ result in a total value in S_{10} that is at least as large as the value of the two summands corresponding to these two indices in S_{01}.

<u>$w = A0B1C$ and $f(A1B0C) > f(X0Y1Z)$:</u> In this case, $A0B1C$ is an index in S_{01}, but $A1B0C$ is not. As $f(A0B1C) \leq f(X0Y1Z) \Rightarrow f(A1B0C) \leq f(X1Y0Z)$, however, in S_{10} not only $u = A0B1C$ is an index, but also $u = A1B0C$ is an index. As $H(A0B1C, X0Y1Z) = H(A1B0C, X1Y0Z)$ and, due to the induction hypothesis, $\mathrm{Prob}(x^{[t-1]} = A0B1C) \leq \mathrm{Prob}(x^{[t-1]} = A1B0C)$, the summand in S_{10} associated with $u = A1B0C$ is at least as large as the summand in S_{01} associated with $w = A0B1C$. (Even more, in contrast to S_{01} where $A1B0C$ is not an index, in S_{10} there is an additional summand associated with $u = A0B1C$.)

All in all, in S_{10} there are as many summands as in S_{01} and they sum to an over-all value at least as large as the value of S_{01}. As seen above, this finishes the induction. □

It is easily seen (using the argument given right after Theorem 1) that this result directly implies Theorem 1, the invariance property of the evolving individual's distribution.

3 On the Probability of Bad Mutations

Now, this result on the distribution of the ones in the evolving individual allows us to obtain better bounds on the drift in the potential, namely the number of 1-bits in the individual. For instance, in the case when a mutation flips two bits x_i and x_j, where $j > i$ (so that $c_j \geq c_i > 0$ for their weights), then $\mathrm{Prob}(x_j = 1 \wedge x_i = 0) \geq \mathrm{Prob}(x_j = 0 \wedge x_i = 1)$. This holds for arbitrarily chosen i, j such that $n \geq j > i \geq 1$. Thus, whenever a mutation flips exactly two bits, then the sub-string flipped is „10“ as least as probable as it is „01.“ For shorter notation, we introduce the following notions:

Definition 1. *Consider the mutation of a given individual $x \in \{0,1\}^n$.*

***B*-mutation.** *Let $B \in \{0,1\}^*$ be the substring in/of x consisting of the bits chosen to be mutated/flipped. Then we call the mutation of x a "B-mutation."*

***z*-zeros-*k*-ones mutation.** *A mutation that flips exactly z zeros and exactly k ones in x.*

***z*-zeros mutation.** *A mutation that flips exactly z zeros (and possibly some ones) in x.*

surely unacceptable mutation. *A mutation that flips <u>more</u> ones than zeros such that each flipping 0-bit can be mapped one-to-one to a flipping 1-bit with a weight at least as large as the weight of the associated zero, respectively.*

potentially acceptable mutation. *A mutation that is not surely unacceptable.*

In the example preceding the definition, we noticed that a 1-zero-1-one mutation is at least as probable a 10-mutation as it is a 01-mutation. More general, since the random choice of the bits to be flipped is independent of the individual ('s distribution over $\{0,1\}^n$), we obtain for our scenario as a direct consequence of Lemma 2 the following:

Corollary 1. *Let $J, K, L \in \{0,1\}^*$. In our scenario the mutation observed in an arbitrary but fixed step is at least as probable a $J1K0L$-mutation as a $J0K1L$-mutation.*

To make actual use of this result, consider the relation $R \subset \cup_{k\in\{2,\dots,n\}}\{0,1\}^k \times \{0,1\}^k$ defined by $(A,B) \in R :\Leftrightarrow (\exists J,K,L \in \{0,1\}^*)\ A = J0K1L \wedge B = J1K0L$. "$(A,B) \in R$" is written as "$A \leq_R B$." Furthermore, we let R^* denote the transitive hull of the relation R. Then the above corollary extends to the following:

Corollary 2. *Let $A, B \in \{0,1\}^k$, $2 \leq k \leq n$, such that $A \leq_{R^*} B$. In our scenario, in an arbitrary but fixed step, a B-mutation occurs at least as probable as an A-mutation.*

Our intermediate objective is to show that, whenever a mutation flips more ones than zeros – which would result in a loss of ones if the mutation was accepted – then this bad mutation is rather likely to be an unacceptable one, so that the loss of ones is *not* accepted. By this, the drift w. r. t. the potential (#ones in the individual) is supposed to become larger, hopefully $\Omega(1/\#$ zeros). The following result will be utilized therefor.

Lemma 3. *In our scenario for $k \geq 2$: Whenever a 1-zero-k-ones mutation occurs in a step, then this mutation is accepted at most with probability $1/(k+1)$.*

Proof. As the bits' weights decrease from left to right, R^* induces a total order on the 1-zero-k-ones mutations because $01^k \leq_R 101^{k-1} \leq_R 1101^{k-2} \leq_R \ldots \leq_R 1^k0$. For $k \geq 2$, a 1-zero-k-ones mutation is potentially acceptable only if it is an 01^k-mutation. The other k of the $\binom{k+1}{1} = k+1$ different mutation types are surely unacceptable. Using the preceding corollary, we know that each of these surely unacceptable types occurs at least as probable as a 01^k-mutation, so that the latter occurs at most with a probability of $1/(k+1)$ (given that a 1-zero-k-ones mutation occurs). □

Analogous results for mutations that flip two (or more) 0-bits can be obtained. For our scenario, however, 1-zero mutations are the crucial ones, so that we focus on these.

4 Better Estimate of the Drift – Better Bound on the Runtime

Let Γ denote the power-set of $\{1, \ldots, n\}$ and $p(b) := p^b(1-p)^{n-b}$ the probability that a mutation flips exactly b bits. Let $\mathrm{Mut}(x, I)$ denote the mutant obtained by flipping the bits in $x \in \{0,1\}^n$ that are determined by the index set $I \in \Gamma$. Then the drift equals

$$\sum_{I \in \Gamma} p(\#I) \cdot \big(\#\{i \in I \mid x_i = 0\} - \#\{i \in I \mid x_i = 1\}\big) \cdot [f(\mathrm{Mut}(x, I)) \geq f(x)]\,, \quad (4)$$

where $[\cdot]$ is an indicator variable that resolves to 1 if the predicate is true, and to 0 otherwise. In the following, the summands will be grouped according to the number $z \in \{0, \ldots, n\}$ of zeros that are flipped. If a z-zeros-k-ones mutation is accepted, the number of ones changes by $z - k$. The probability that exactly z zeros and k ones are flipped in $x \in \{0,1\}^n$ equals $\boldsymbol{P_{x,p}(z,k)} := \binom{|x|_0}{z} \cdot \binom{|x|_1}{k} \cdot p^{z+k} \cdot (1-p)^{n-(z+k)}$. For $k > z \geq 1$, let $\boldsymbol{A_{x,p}(z,k)}$ denote an upper bound on the probability that a z-zeros-k-ones mutation of $x \in \{0,1\}^n$ is accepted. Let $z \geq 1$ be fixed. For $k > z$, the summands in Eqn. (4) for which I corresponds to a z-zeros-k-ones mutation sum up to a negative value not smaller than $P_{x,p}(z,k) \cdot A_{x,p}(z,k) \cdot (z-k)$. A z-zero-0-ones mutation is always accepted and increases the number of ones by z (≥ 1). All summands for which I corresponds to a z-zeros-0-ones mutation sum up to $P_{x,p}(z,0) \cdot z$. Thus, for the fixed number $z \geq 1$ of flipping zeros, the contribution of all possible z-zero mutations to the drift (the expected change in the number of 1-bits) is at least

$$\Delta_{x,p}(z) := P_{x,p}(z,0) \cdot z + \textstyle\sum_{z<k\leq|x|_1} P_{x,p}(z,k) \cdot A_{x,p}(z,k) \cdot (z-k)\,. \quad (5)$$

Up to now, $z \geq 1$ was assumed. For $z = 0$, note that in our scenario a mutation that does not flip a 0-bit cannot change the individual, so that the total drift is bounded below by $\boldsymbol{\Delta_{x,p}} := \sum_{z=1}^{|x|_0} \Delta_{x,p}(z)$. For $z \geq 1$ the formula for $\Delta_{x,p}(z)$ can be transformed into

$$\begin{aligned} & z \cdot \Big(P(z,0) + \sum_{z<k\leq|x|_1} P(z,k) \cdot A(z,k) \cdot \frac{z-k}{z} \Big) \\ &= z \cdot \binom{|x|_0}{z} \cdot \Big(p^z \cdot (1-p)^{n-z} + \sum_{z<k\leq|x|_1} \binom{|x|_1}{k} \cdot p^{z+k} \cdot (1-p)^{n-(z+k)} \cdot A(z,k) \cdot \frac{z-k}{z} \Big) \\ &= z \cdot \binom{|x|_0}{z} \cdot p^z (1-p)^{n-z} \cdot \Big(1 + \sum_{z<k\leq|x|_1} \tbinom{|x|_1}{k} \cdot \Big(\frac{p}{1-p} \Big)^k \cdot A(z,k) \cdot \frac{z-k}{z} \Big) . \end{aligned} \tag{6}$$

The sign of $\Delta_{x,p}(z)$ is determined by the sign of the rightmost factor (in big parentheses). We now show that $\boldsymbol{\Delta_x(z)} := \Delta_{x,1/n}(z)$ is non-negative for all z when $p = 1/n$.

Lemma 4. *In our scenario, where the mutation rate $p = 1/n$ is used, for any fixed individual $x \in \{0,1\}^n$: $\Delta_x(z) \geq 0$ for $z \in \{1, \ldots, |x|_0\}$.*

Proof. Obviously, $z \cdot \binom{|x|_0}{z} \cdot p^z(1-p)^{n-z} \geq 0$, so that we have to show that the rightmost factor $(1 + \sum \ldots)$ in Eqn. (6) is non-negative. $A(z,k) := 1$ is a trivial upper bound on the probability that a (bad) mutation is accepted, so that we concentrate on

$$\sum_{z<k\leq|x|_1} \binom{|x|_1}{k} \cdot \Big(\frac{p}{1-p} \Big)^k \cdot \frac{-(z-k)}{z} \leq 1. \tag{7}$$

We take a closer look at the summands: $\left(\frac{p}{1-p}\right)^k = \left(\frac{1}{n\cdot(1-1/n)}\right)^k = (n-1)^{-k}$ and $\binom{|x|_1}{k} \leq \frac{(n-1)^k}{k!}$ since $|x|_1 =\leq n-1$. Thus, the sum in Eqn. (7) is bounded above by

$$\sum_{z<k\leq|x|_1} \frac{(n-1)^k}{k!} \cdot (n-1)^{-k} \cdot \frac{k-z}{z} = \sum_{z<k\leq|x|_1} \frac{k-z}{k! \cdot z} \leq \sum_{k\geq 2} \frac{k-1}{k!} \tag{8}$$

since $z \geq 1$ is assumed. The rightmost sum equals $\sum_{k\geq2} \left(\frac{1}{(k-1)!} - \frac{1}{k!} \right) = \frac{1}{(2-1)!} = 1$, which proves the inequality Eqn. (7) and with it the claimed inequality $\Delta_x(z) \geq 0$. □

So, now we know that $\Delta_x(z)$ is non-negative for any number z of zeros that may be flipped by a mutation – whatever the mutated individual x. Consequently, we know that throughout the optimization process the drift is non-negative in each iteration. For a good upper bound on the runtime, however, we need a positive lower bound on the drift. In fact, the larger the lower bound on the drift, the better. Since for $p = 1/n$ the expected number of bits that flip equals one, we take a second look at $\Delta_x(1)$ to derive a better estimation for the contribution of 1-zero mutations to the drift. In Eqn. (6) the sum over k has been estimated for $z = 1$ using the trivial estimate $A_{x,p}(z,k) = 1$. Now, making use of our knowledge about the distribution of the evolving individual x over $\{0,1\}^n$, namely by Lemma 3, we know that for $k \geq 2$ a 1-zero-k-ones mutation is accepted at most with probability $1/(k+1)$, so that for our scenario $A(1,k) := 1/(k+1)$ can be used instead of the distribution-independent/trivial upper bound $A_{x,p}(1,k) = 1$.

Lemma 5. *For* $p := 1/n$, $A(1,k) := 1/(k+1)$, $z := 1$ *the following inequality holds:* $\sum_{z<k\leq n} \binom{|x|_1}{k} \cdot \left(\frac{p}{1-p}\right)^k \cdot A(z,k) \cdot \frac{k-z}{z} < 0.282$.

Proof. As in proof of the previous lemma, the sum to be bounded from above is at most $\sum_{z<k\leq n} \frac{1}{k!} \cdot A(z,k) \cdot \frac{k-z}{z}$. For the settings of the lemma we obtain

$$\begin{aligned}\sum_{z<k\leq n} \frac{1}{k!} \cdot A(z,k) \cdot \frac{k-z}{z} &\leq \sum_{k\geq 2} \frac{1}{k!} \cdot \frac{1}{k+1} \cdot (k-1) \;=\; \sum_{k\geq 2} \frac{k-1}{(k+1)!} \\ &= \sum_{k\geq 3} \frac{k-2}{k!} = \sum_{k\geq 2} \frac{1}{k!} - \sum_{k\geq 3} \frac{2}{k!} \;=\; (\mathrm{e}-2) - 2(\mathrm{e}-2.5)\end{aligned}$$

using $\sum_{k\geq 1} 1/k! = \mathrm{e} - 1$. Finally, $(\mathrm{e}-2) - 2(\mathrm{e}-2.5) \;=\; 3 - \mathrm{e} \;<\; 0.282$. □

Plugging the estimate of the preceding Lemma 5 into Eqn. (6) for $z := 1$, we obtain

$$\Delta(1) \;\geq\; |x|_0 \cdot \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \cdot (1 - 0.282) \;\geq\; \frac{|x|_0}{n} \cdot \frac{0.718}{\mathrm{e}} \;>\; \frac{|x|_0}{n} \cdot 0.264$$

(Recall that $\Delta(z) \geq 0$ for all z.) This better estimation of the contribution of 1-zero mutations to the drift utilizing the individual's distribution over the search space $\{0,1\}^n$ is crucial: Now the lower bound on the drift is $\Delta \geq \Delta(1) \geq \frac{|x|_0}{n} \cdot 0.264 = \Omega(\#\text{zeros}/n)$. When we consider the number of zeros as the approximation error (the Hamming distance from the optimum), then the result on the drift reads: As long as there are zeros in x, in each step we expect the approximation error to decrease by a factor smaller (i. e. better) than $1 - 0.264/n$. Since $(1 - 0.264/n)^{(n/0.264)\cdot \ln 2} \lesssim 1/2$, the number of steps until we expect the progress to be such that the approximation error is halved is less than $(n/0.264) \cdot \ln 2 < 2.63n$. The actual question is, however: What is the expected number of steps to actually halve the approximation error? The following lemma helps us to turn our lower bound on the drift into an upper bound on the expected runtime:

Lemma 6. *Let* $X_1, X_2, \ldots$ *denote random variables with bounded support and* S *the random variable defined by* $S := \min\{\, t \mid X_1 + \cdots + X_t \geq g\}$ *for a given* $g > 0$. *Given that* S *is a stopping time (i. e., the event* $\{S = k\}$ *depends solely on* $X_1, \ldots, X_k$*), if* $\mathsf{E}[S] < \infty$ *and* $\mathsf{E}[X_i \mid S \geq i\,] \geq \ell > 0$ *for all* i*, then* $\mathsf{E}[S] \leq \mathsf{E}[X_1 + \cdots + X_S]/\ell$.

Proof. Note that the X_i need not be independent and that, since the X_i are bounded, the precondition $\mathsf{E}[S] < \infty$ implies $\mathsf{E}[X_1 + \cdots + X_S] < \infty$. We use

$$\mathsf{E}[X_1 + \cdots + X_S] = \sum_{i=1}^{\infty} \mathrm{Prob}\{S \geq i\,\} \cdot \mathsf{E}[X_i | S \geq i\,] \geq \sum_{i=1}^{\infty} \mathrm{Prob}\{S \geq i\,\} \cdot \ell = \mathsf{E}[S] \cdot \ell$$

where the first equation is the major part of the proof of Wald's equation (a proof can be found in [6, Apx. B] for instance). □

We concentrate on the expected number of steps to halve the approximation error, and thus, in the application of the preceding lemma we let X_i denote the increase in the number of ones in the ith iteration and choose $g := z/2$ and $\ell := (z/2) \cdot 0.264/n$, where we use that $0 \le X_i \le n$ in our scenario, and that the condition $\{S \ge i\}$ merely means that the approximation error has not been halved within the first $i-1$ iterations. Finally, we use $\mathsf{E}[X_1 + \cdots + X_S] \le z/2 + z/n$, where "$+z/n$" is a rough general upper bound on the expected increase in the number of ones (in a step), namely the expected number of flipping zeros, which is $z \cdot p = z/n$ since $p = 1/n$. Thus, the application of the previous lemma yields the following upper bound on the expectation of the number S of steps to halve the approximation error, i. e. the number of 0-bits:

$$\mathsf{E}[S] \le \frac{\mathsf{E}[X_1 + \cdots + X_S]}{\ell} \le \frac{z/2 + z/n}{(z/2) \cdot 0.264/n} \le 3.79n + 7.58.$$

With this bound on the zeros' expected half-life we can finally derive our main result.

Theorem 2. *Let the (1+1) EA using the mutation rate (bit-flip probability)* $1/n$ *maximize a linear function* $f\colon \{0,1\}^n \to \mathbb{R}$, $n \ge 2$. *Then the expected number of steps until the evolving individual has maximum* f*-value is smaller than* $3.8\, n \log_2 n + 7.6 \log_2 n$.

Proof. Without loss of generality we may assume that all coefficients are positive, so that the all-ones string has maximum function value. As the expected number of 0-bits in the initial individual equals $n/2$, after $\lfloor \log_2(n/2) \rfloor + 1 \le \log_2 n$ halvings (in expectation w. r. t. the initialization) there is less than one 0-bit left, i. e., the optimal all-ones bit-string has been generated. Since the expected number of iterations to halve the number of zeros is smaller than $3.8n + 7.6$ (independently of the initialization), we obtain an upper bound of $(3.8n + 7.6) \cdot \log_2 n$ on the expected number of iterations. □

5 Conclusions

We have exemplarily shown for the fundamental scenario "standard (1+1) EA on linear functions" how knowledge about the distribution of the evolving individual over the search space can significantly improve (upper) bounds on the expected runtime until an optimum is generated. The invariance property of the individual's distribution proved and then utilized here is actually quite intuitive. Interestingly, its proof is quite straightforward and not that sophisticated. Nonetheless, it enables a remarkable improvement of the estimation of the drift (and, thus, of the runtime). For other scenarios, a suitable invariance property may be harder to find – experiments may actually give useful hints – and probably even harder to prove. But, as we have demonstrated here, these efforts may indeed pay off. Actually, in some situations such knowledge is necessary to obtain an asymptotically tight bound. Then the use of an invariance property of the evolving individual's distribution over the search space can actually be an elegant tool.

Naturally, for more advanced (evolutionary) algorithms that use a population, the distribution of the population over the search space may be considered. In [7] a similar approach is followed for the analysis of a $(\mu+1)$ evolution strategy, showing that this technique can indeed make sense for the analysis of population-based algorithms.

References

1. Rudolph, G.: Finite Markov chain results in evolutionary computation: A tour d'horizon. Fundamenta Informaticae 35, 67–89 (1998)
2. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–82 (2002)
3. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence 127, 57–85 (2001)
4. He, J., Yao, X.: Erratum to: Drift analysis and average time complexity of evolutionary algorithms [3]. Artificial Intelligence 140, 245–248 (2002)
5. Jägersküpper, J.: A mix of Markov-chain and drift analysis. Technical Report CI-250/08, TU Dortmund, SFB 531 (2008)
6. Jägersküpper, J.: Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. Theoretical Computer Science 379, 329–347 (2007)
7. Jägersküpper, J., Witt, C.: Rigorous runtime analysis of a (μ+1) ES for the Sphere function. In: Proc. 2005 Genetic and Evolutionary Computation Conference (GECCO), pp. 849–856. ACM Press, New York (2005)

On Multiplicative Noise Models for Stochastic Search

Mohamed Jebalia[1] and Anne Auger[1,2]

[1] TAO Team, INRIA Saclay, Université Paris Sud, LRI, 91405 Orsay cedex, France
[2] Microsoft Research-INRIA Joint Centre 28, rue Jean Rostand, 91893 Orsay Cedex, France
mohamed.jebalia@lri.fr, anne.auger@inria.fr

Abstract. In this paper we investigate multiplicative noise models in the context of continuous optimization. We illustrate how some intrinsic properties of the noise model imply the failure of reasonable search algorithms for locating the optimum of the noiseless part of the objective function. Those findings are rigorously investigated on the $(1+1)$-ES for the minimization of the noisy sphere function. Assuming a lower bound on the support of the noise distribution, we prove that the $(1+1)$-ES diverges when the lower bound allows to sample negative fitness with positive probability and converges in the opposite case. We provide a discussion on the practical applications and non applications of those outcomes and explain the differences with previous results obtained in the limit of infinite search-space dimensionality.

1 Introduction

In many real-world optimization problems, objective functions are perturbed by noise. Evolutionary Algorithms (EAs) have been proposed as effective search methods in such contexts [5,10]. A noisy optimization problem is a rather general optimization problem where for each point x of the search space, we can observe $f(x)$ perturbed by a random variable or in other words for a given x we can observe a distribution of possible objective values. The goal is in general to converge to the minimum of the averaged value of the observed random variable. One type of noise encountered in real-world problems is the so-called multiplicative noise where the noiseless objective function $f(x)$ is perturbed by the addition of a noise term proportional to f, ie. the noisy objective function $\mathcal{F}$ reads

$$\mathcal{F}(x) = f(x)(1+\mathcal{N}) \tag{1}$$

where $\mathcal{N}$ is the noise random variable, sampled independently at each new evaluation of a point. Such noise models are in particular used to benchmark robustness of EAs with respect to noise [12]. The focus here is continuous optimization (that will be minimization) where f maps a continuous search space, ie. a subset of $\mathbb{R}^d$, into $\mathbb{R}$. The EAs specifically designed for continuous optimization are usually referred as Evolution Strategies (ES), where a set of candidate solutions evolves by first applying Gaussian perturbations (mutations) to the current solutions then selection. ES in noisy environments have been studied by Arnold and Beyer [8,3,1]. Multiplicative noise has been investigated in the case of $\mathcal{N}$ being normally distributed with a standard deviation scaled by $1/d$ for a $(1+1)$-ES [4], (μ,λ)-ES [3,7], $(\mu/\mu_I,\lambda)$-ES [2] and f being the sphere function $f(x) = \|x\|^2$. Under the assumption that d goes to infinity, Arnold and

Beyer show, for $f(x) = \|x\|^2$, positive expected fitness gain for the elitist $(1+1)$-ES (if the fitness of the parent is not reevaluated in the selection step which is the case of our study). This implies a decrease of the expectation of the square distance to the optimum (here zero). However, convergence of the $(1 + 1)$-ES to the optimum of the noiseless part of the noisy objective function seems to be unlikely if the noise random variable takes values smaller than -1 as we illustrate now on a simple example. Assume indeed that $\mathcal{N}$ takes three distinct values (each with probability $1/3$) $+\gamma$, 0 and $-\gamma$ where γ satisfies $\gamma > 1$. For a given $x \in \mathbb{R}^d$, the objective function $\mathcal{F}(x)$ takes 3 different values (each with probability $1/3$) $(1 + \gamma)\|x\|^2$, $\|x\|^2$, $(1 - \gamma)\|x\|^2$. The last term is strictly negative for x non equal to zero. Therefore, if one negative objective function value is reached, the $(1+1)$-ES that can only accept solutions having a lower objective function value will never accept solutions closer to the optimum since they have higher objective function values[1]. On the contrary the $(1 + 1)$-ES will diverge log-linearly[2], i.e. the logarithm of the distance to the optimum will increase linearly.

Starting from this observation, we investigate how the properties of the support of the noise distribution relate to convergence or divergence of stochastic search algorithms and can make the convergence to the optimum of the noiseless part of the objective function hopeless for reasonable search algorithms. Compared to previous approaches, we do not make use of asymptotic assumptions, trying to capture effects that were not observed before [4]. In Section 2, we detail the noise model considered and show experimentally on a $(1 + 1)$-ES that divergence and convergence is determined by the probability to sample noise values smaller than -1. In Section 3, we provide some simple proofs of convergence and divergence for the $(1+1)$-ES. In Section 4 we discuss the results and explain where the difference with the results in [4] stems from.

2 Motivations

Elementary Remarks on the Noise Model. We investigate multiplicative noise models as defined in Eq. 1 where $\mathcal{N}$ is a random variable with finite mean and $f(x)$ is the noiseless function that we assume positive in the sequel. We also assume that $1 + E(\mathcal{N}) > 0$ such that the argmin[3] of the expected value of $\mathcal{F}(x)$ is the argmin of $f(x)$. Often, the distribution of $\mathcal{N}$ is assumed symmetric, implying then that $1 + E(\mathcal{N}) = 1 > 0$. Though one might think that this condition is sufficient such that minimizing $\mathcal{F}(x)$ amounts to minimizing $f(x)$, we sketch now, why divergence to ∞ of the distance to the optimum happens if $1 + \mathcal{N}$ can take negative values.

Assume that $f(x)$ converges to infinity when $\|x\|$ goes to ∞; typically $f(x)$ can be the famous sphere function $f(x) = \|x\|^2$ and assume that the random variable $\mathcal{N}$ admits a density function $p_{\mathcal{N}}(t), t \in \mathbb{R}$ whose support is an interval $[m_{\mathcal{N}}, M_{\mathcal{N}}[$, i.e. $\mathcal{N} \in [m_{\mathcal{N}}, M_{\mathcal{N}}[$ and the probability that $\mathcal{N} \in [a, b]$ for any $m_{\mathcal{N}} \leq a < b \leq M_{\mathcal{N}}$ is

[1] Their absolute value is smaller though. However, trying to minimize the absolute value of $\mathcal{F}$ instead is not a solution in general, consider for instance the function $f(x) = (\|x\|^2 + 1)(1 + \mathcal{N})$.

[2] We will say that a sequence $(d_n)_n$ diverges (resp. converges) log-linearly if there exists $c > 0$ (resp. $c < 0$) such that $\lim_n \frac{1}{n} \ln(d_n) = c$.

[3] The argmin of an objective function $x \mapsto h(x)$ are defined as $h(\arg\min_x h) = \min_x h(x)$.

strictly positive. The function $g_{m_\mathcal{N}}(x) = f(x)(1 + m_\mathcal{N})$ gives a lower bound of the values that can be reached by the noisy fitness function for different instantiations of the random variable $\mathcal{N}$ (because f is positive). For a given x, $\mathcal{F}(x)$ can take values with positive probability in any open interval of $]g_{m_\mathcal{N}}(x), f(x)[$ [4].

In Fig. 1 are depicted a cut of $f(x) = \|x\|^2$ and $g_{m_\mathcal{N}}(x) = f(x)(1 + m_\mathcal{N})$ for $m_\mathcal{N}$ equals -0.5 and -1.5. The position of $m_\mathcal{N}$ with respect to -1 determines whether $g_{m_\mathcal{N}}(x)$ is convex or concave: for $m_\mathcal{N} > -1$, $g_{m_\mathcal{N}}(x)$ is convex, converging to infinity when $\|x\|$ goes to ∞ and for $m_\mathcal{N} < -1$, $g_{m_\mathcal{N}}(x)$ is concave, converging to minus infinity when $\|x\|$ goes to ∞. Minimizing $g_{m_\mathcal{N}}(x)$ in the case of $m_\mathcal{N} < -1$ means that $\|x\|$ is diverging to $+\infty$ and $g_{m_\mathcal{N}}(x)$ is diverging to $-\infty$ which is the opposite of the behavior one would like since we are aiming at minimizing the non-noisy function $f(x) = \|x\|^2$. Note that in the example sketched in the introduction with $\mathcal{N}$ taking

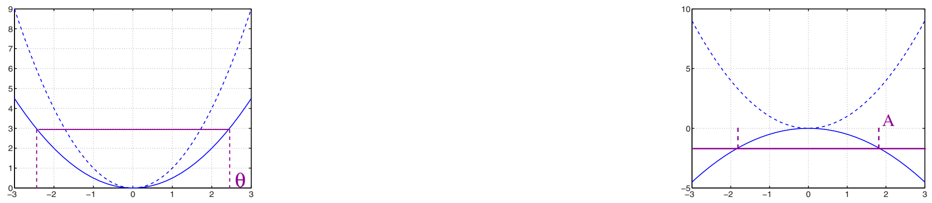


Fig. 1. [Dashed Line] One dimensional cut of $f(x) = \|x\|^2$ along one arbitrary unit vector. [Straight line] Left: One dimensional cut of $g_{-0.5}(x) = \|x\|^2(1 - 0.5)$. Right: One dimensional cut of $g_{-1.5}(x) = \|x\|^2(1 - 1.5)$. For a given x, the noisy-objective function can, in particular, take any value between the dashed curve and the straight curve.

the values γ, $-\gamma$ and 0, the plot of $\|x\|^2$ and $(1 - \gamma)\|x\|^2$ for $\gamma = 1.5$ are the curves represented in Fig 1 (right).

Experimental Observations. We investigate now numerically how the "shape" of the lower bound might affect the convergence. For this purpose we use a $(1, 5)$-ES and a $(1 + 1)$-ES using scale-invariant adaptation scheme for the step-size[5].

We investigate the function $\mathcal{F}_s(x) = \|x\|^2(1 + \mathcal{N})$ when the noise $\mathcal{N}$ is uniformly distributed in the ranges $[-0.5, 0.5]$ and $[-1.5, 1.5]$ respecitvely denoted $U_{[-0.5,0.5]}$ and $U_{[-1.5,1.5]}$. This latter noise corresponds to the concave lower bound $g_{-1.5}(x) = -0.5\|x\|^2$ plotted in Fig. 1. In Figure 2, the result of 10 independent runs of the $(1, 5)$-ES (10 upper curves of each graph) in dimension $d = 10$ are plotted for the non-noisy sphere (left), $f(x) = \|x\|^2(1 + U_{[-0.5,0.5]})$ (middle) and $f(x) = \|x\|^2(1 + U_{[-1.5,1.5]})$ (right). Not too surprisingly, we observe a drastic difference in the last two cases: the algorithm converges to the optimum for the noise $U_{[-0.5,0.5]}$ whereas the distance to the

[4] Note that $g_{m_\mathcal{N}}(x) < f(x)$ iff $m_\mathcal{N} < 0$.

[5] In a scale-invariant ES, the step-size is set at each iteration as a (strictly positive) constant σ times the distance to the optimum. This artificial adaption scheme (since in practice one does not know the distance to the optimum!) allows to achieve optimal convergence rate for ES and is therefore very interesting from a theoretical point of view. The algorithm is mathematically defined in Section 3.

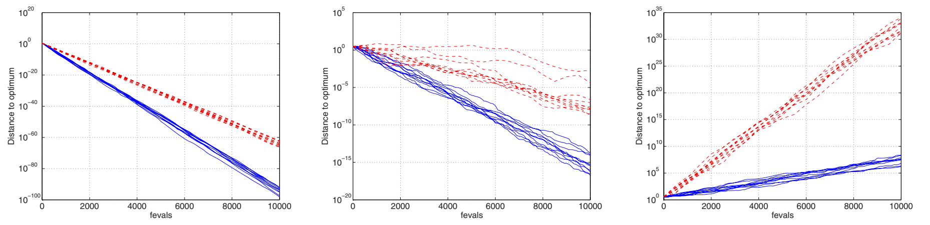


Fig. 2. Distance to the optimum (in log-scale) versus number of evaluations. Ten independent runs for the scale-invariant $(1, 5)$-ES (10 upper curves of each graph) and $(1 + 1)$-ES (10 lower curves of each graphs) with $d = 10$ and $\sigma = 1/d$. Left: $f(x) = \|x\|^2$. Middle: $f(x) = \|x\|^2(1+ U_{[-0.5,0.5]})$. Right: $f(x) = \|x\|^2(1 + U_{[-1.5,1.5]})$.

optimum increases (log)-linearly for the noise having a lower bound smaller than -1[6]. Comparing the left and middle graphs we also observe, as expected, that the presence of noise slows down the convergence. On the same figure (lower curves of the graphs), the results of 10 independent runs of the $(1+1)$-ES are plotted for the three same functions. As in the case of the comma strategy we observe that the $(1 + 1)$-ES diverges in the case of the noise $U_{[-1.5,1.5]}$ and that, when convergence occurs, the convergence rate is slower in presence of noise. Last, we investigate numerically the $(1 + 1)$-ES where $\mathcal{N}$ is normally distributed and in particular unbounded. This corresponds to the case investigated in [4]. We carry out tests for a standard deviation of the Gaussian noise equals 0.1, 2 and 10. Results are presented in Fig. 3. We observe convergence when the standard deviation of the noise equals 0.1 and divergence in the last two cases.

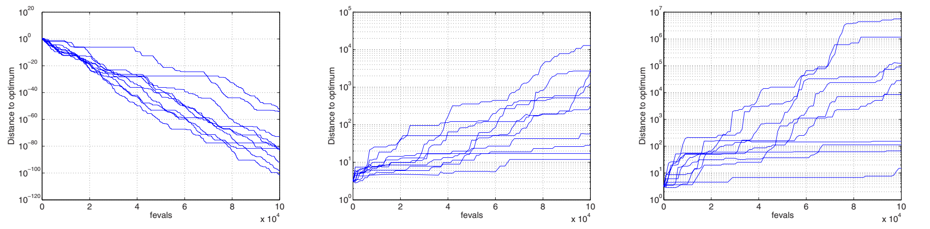


Fig. 3. Ten independent runs for the scale-invariant $(1 + 1)$-ES with a normally distributed noise: on $f(x) = \|x\|^2(1 + \sigma_\epsilon\mathcal{N}(0, 1))$ with σ_ϵ equals 0.1 (left), 2 (middle) and 10 (right) for $d = 10$ and $\sigma = 1/d$

3 Convergence and Divergence of the $(1 + 1)$-ES

In this section, we provide a simple mathematical analysis of the convergence and divergence of the $(1 + 1)$-ES experimentally observed in the previous section. We focus for

[6] However, contrary to what we will see for the $(1 + 1)$-ES, we do not state that "-1" is a limit value between convergence and divergence in the case of $(1, \lambda)$-ES. Indeed convergence and divergence depends on the intrinsic properties of the noise and on λ and σ as well (see [8]).

the sake of simplicity on lower bounded noise, i.e. the support of the noise is included in $[m_{\mathcal{N}}, +\infty[$. We prove that the $(1+1)$-ES minimizing the noisy sphere converges if $m_{\mathcal{N}} > -1$ and diverges if $m_{\mathcal{N}} < -1$. The proofs are rather simple and rely on the Borel-Cantelli Lemma. For the sake of readability we provide here a sketch of the demonstrations and send the proofs with the technical details in the Appendix of the paper.

Mathematical Model for the $(1+1)$-ES. The $(1+1)$-ES is a simple ES which evolves a single solution. At an iteration n, this solution denoted X_n, is called parent. The minimization of a given function f mapping $\mathbb{R}^d$ $(d \geq 1)$ into $\mathbb{R}$ using the $(1+1)$-ES algorithm is as follows: At every iteration n, the parent X_n is perturbated by a Gaussian random variable $\sigma_n \mathrm{N}_n$, where σ_n is a strictly positive value called step-size and $(\mathrm{N}_n)_n \in \mathbb{R}^d$ are independent realizations of a multivariate isotropic normal distribution on $\mathbb{R}^d$ denoted by $\mathrm{N}(0, I_d)$ [7]. The resulting offspring $\mathrm{X}_n + \sigma_n \mathrm{N}_n$ is accepted if and only if its fitness value is smaller than the one of its parent X_n. One of the key points in minimization using isotropic ES[8] is how to adapt the sequence of step-sizes (σ_n). Convergence of the $(1+1)$-ES is sub-log-linear bounded below by an explicit log-linear rate. This lower bound for the convergence rate is attained for the specific case of the sphere function and scale-invariant algorithm where the step-size is chosen proportional to the distance to the optimum, i.e. $\sigma_n = \sigma \|\mathrm{X}_n\|$ where σ is a strictly positive constant [6,9]. The scale-invariant algorithm has a major place in the theory of ES since it corresponds to the dynamic algorithm implicitly studied in the one-step analysis computing progress rate or fitness gain [11,8]. Using this adaptation scheme, the algorithm is referred to as the scale-invariant $(1+1)$-ES and the offspring writes as $\mathrm{X}_n + \sigma \|\mathrm{X}_n\| \mathrm{N}_n$. The noisy sphere function is denoted

$$\mathcal{F}_s(x) = \|x\|^2 (1 + \mathcal{N}) \tag{2}$$

where we assume that the random variable $\mathcal{N}$ has a finite expectation such that $E(\mathcal{N}) > -1$ and admits a density function $p_{\mathcal{N}}$ which lies in the range $[m_{\mathcal{N}}, M_{\mathcal{N}}[$ where $-\infty < m_{\mathcal{N}} < M_{\mathcal{N}} \leq +\infty$, $M_{\mathcal{N}} > -1$ and $m_{\mathcal{N}} \neq -1$. The normalized noisy part $\mathcal{N}$ of the noisy sphere function will be called normalized overvaluation of x. The term normalized overvaluation was already defined in [4] where it corresponds to the opposite of the quantity considered here up to a factor $d/2$. The minimization of this function using the scale-invariant $(1+1)$-ES is mathematically modeled by the sequence of parents (X_n) with their relative noisy fitnesses $(\mathcal{F}_s(\mathrm{X}_n))$ and normalized overvaluations (O_n). At an iteration n, the fitness of the parent is $\mathcal{F}_s(\mathrm{X}_n) = \|\mathrm{X}_n\|^2 (1 + O_n)$ and the fitness of an offspring equals $\|\mathrm{X}_n + \sigma \|\mathrm{X}_n\| \mathrm{N}_n\|^2 (1 + \mathcal{N}_n)$ where $(\mathcal{N}_n)_n$ is a sequence of independent random variables with $\mathcal{N}$ as a common law. Let $\mathrm{X}_0 \in \mathbb{R}^d$ be the first parent with a normalized overvaluation O_0 sampled from the distribution of $\mathcal{N}$. Then the update of X_n for $n \geq 0$ writes as:

$$\begin{aligned} \mathrm{X}_{n+1} &= \mathrm{X}_n + \sigma \|\mathrm{X}_n\| \mathrm{N}_n \text{ if } \|\mathrm{X}_n + \sigma \|\mathrm{X}_n\| \mathrm{N}_n\|^2 (1 + \mathcal{N}_n) < \|\mathrm{X}_n\|^2 (1 + O_n)\,, \\ &= \mathrm{X}_n \text{ otherwise}\,, \end{aligned} \tag{3}$$

[7] $\mathrm{N}(0, I_d)$ is the multivariate normal distribution with mean $(0, \ldots, 0) \in \mathbb{R}^d$ and covariance matrix the identity I_d.

[8] ES are called isotropic when the covariance matrix of the distribution of the random vectors $(\mathrm{N}_n)_n$ is I_d.

and the new normalized overvaluation O_{n+1} is then:

$$\begin{aligned} O_{n+1} &= \mathcal{N}_n \text{ if } \|\mathrm{X}_n + \sigma\|\mathrm{X}_n\|\mathrm{N}_n\|^2 (1+\mathcal{N}_n) < \|\mathrm{X}_n\|^2 (1+O_n) , \\ &= O_n \text{ otherwise} . \end{aligned} \tag{4}$$

The $(1+1)$-ES algorithm ensures that the sequence relative to the function to minimize (which is $(\mathcal{F}_s(\mathrm{X}_n))$ in our case) decreases. This property makes the theoretical study of the $(1+1)$-ES easier than that of comma strategies. Our study shows that the behavior of the scale-invariant $(1+1)$-ES on the noisy sphere function (2) depends on the lower bound of the noise $m_{\mathcal{N}}$.

Theorem 1. *The* $(1+1)$*-ES minimizing the noisy sphere (Eq. 2) defined in Eq. 3 converges to zero if* $m_{\mathcal{N}} > -1$ *and diverges to infinity when* $m_{\mathcal{N}} < -1$.

Proof. The proof of this theorem is split in two cases $m_{\mathcal{N}} > -1$ and $m_{\mathcal{N}} < -1$ respectively investigated in Proposition 1 and Proposition 2. □

The proofs heavily rely on the second Borel-Cantelli Lemma that we recall below. But first, we need a formal definition of 'infinitely often (i.o.)': Let q_n be some statement, eg. $|a_n - a| > \epsilon$. We say (q_n i.o.) if for all n, $\exists\, m \geq n$ such that q_m is true. Similarly, for a sequence of events A_n in a probability space, (A_n i.o.) equals $\{w | w \in A_n \text{ i.o.}\} = \cap_{n\geq 0} \cup_{m \geq n} A_m := \overline{\lim}\, A_n$. The second Borel-Cantelli Lemma (BCL) states that:

Lemma 1. *Let* $(A_n)_{n\geq 0}$ *be a sequence of events in some probability space. If the events* A_n *are independent and verify* $\sum_{n\geq 0} P(A_n) = +\infty$ *then* $P(\overline{\lim}\, A_n) = 1$.

Proposition 1 (Convergence for $m_{\mathcal{N}} > -1$**).** *If* $m_{\mathcal{N}} > -1$*, the sequences* $(\mathcal{F}_s(\mathrm{X}_n))$ *and* $(\|\mathrm{X}_n\|)$ *converge to zero almost surely.*

Sketch of the proof (see detailed proof in Appendix) The condition $m_{\mathcal{N}} > -1$ ensures that the decreasing sequence $(\mathcal{F}_s(\mathrm{X}_n))$ is positive. Therefore it converges. Besides the sequence $(\|\mathrm{X}_n\|)$ is upper bounded by $\theta := \mathcal{F}_s(X_0)/(1+m_{\mathcal{N}})$ as shown in Fig. 1 (left). Consequently, the probability to hit, at each iteration n, a fixed neighborhood of 0 is lower bounded by a strictly positive constant. Applying BCL we deduce the convergence of the sequence $(\mathcal{F}_s(\mathrm{X}_n))$ (and then that of $(\|\mathrm{X}_n\|)$) to zero. □

Proposition 2 (Divergence for $m_{\mathcal{N}} < -1$**).** *If* $m_{\mathcal{N}} < -1$*, the sequence* $(\mathcal{F}_s(\mathrm{X}_n))$ *diverges to* $-\infty$ *almost surely and the sequence* $(\|\mathrm{X}_n\|)$ *diverges to* $+\infty$ *almost surely.*

Sketch of the proof (see detailed proof in Appendix) As $1 + m_{\mathcal{N}} < 0$, the probability to sample a noise $\mathcal{N}_n$ such that $1 + \mathcal{N}_n < 0$ is striclty positive. Therefore there exists an integer n_1 such that for all $n \geq n_1$, $\mathcal{F}_s(\mathrm{X}_n) < 0$. Consequently $(\|\mathrm{X}_n\|)$ is lower bounded by A as illustrated in Fig. 1 (right) where the straight horizontal line represents the slope $y = \mathcal{F}_s(X_{n_1})$. Besides, the probability to have $\mathcal{F}_s(\mathrm{X}_n)$ as small as we want is lower bounded by a strictly positive constant which gives with BCL the divergence of the sequence $(\mathcal{F}_s(\mathrm{X}_n))$ to $-\infty$, i.e. the sequence $(\|\mathrm{X}_n\|)$ diverges to $+\infty$. □

Remark that for the example sketched in the introduction where $\mathcal{N}$ takes the 3 different values γ, 0 and $-\gamma$ and under the condition $\gamma > 1$ the proof of divergence will follow the same lines.

4 Discussion and Conclusion

We conclude from Theorem 1 that what matters for convergence or divergence of the $(1+1)$-ES in the case of noisy objective function with positive noiseless part is the position of the lower bound $m_{\mathcal{N}}$ of the noise distribution $\mathcal{N}$ with respect to -1 or in other words the existence or not of possible negative fitness values. This result applies in particular when $\mathcal{N}$ equals a truncated normal distribution, i.e. $\mathcal{N} = \sigma_\epsilon \mathcal{N}(0,1)1_{[-a,a]}$[9] for any a and σ_ϵ positive. Whenever $\sigma_\epsilon a > 1$, Proposition 2 applies and the $(1+1)$-ES diverges.

Those results might appear in contradiction with those of Arnold and Beyer [4] proving that the expected fitness gain is positive$-$and therefore convergence in mean holds for the scale-invariant ES$-$for a noise distributed according to a normal distribution. In their model, Arnold and Beyer scale the standard deviation of the noise σ_ϵ with $1/d$, i.e. when $d \to \infty$, σ_ϵ converges to 0. The largest value for the normalized σ_ϵ^* in [4, Fig 5, 6, 8], for $d = 80$ corresponds to a standard deviation of 0.05 for which the probability to have $(1 + 0.05\,\mathcal{N}) < 0$ is upper bounded by 10^{-88} [10], i.e. relatively unlikely! Therefore though they consider some unbounded noise having a support in $\mathbb{R}$, the normalization of the standard deviation of the noise implies a so small probability to sample $1+\mathcal{N}$ below -1 that the unbounded noise reduces to the case of convergence where $m_{\mathcal{N}} > -1$. The same conclusion holds for the numerical example given in Section 2, Fig. 3 (left) where the standard deviation of 0.1 corresponds to a probability to have $(1+0.1\,\mathcal{N}) < 0$ lower bounded by 10^{-23}. Therefore though the theory predicts divergence as soon as $m_{\mathcal{N}} < -1$, what matters in practice is how likely the probability to sample $\mathcal{N} < -1$ is.

In conclusion, we have illustrated that convergence but also divergence can happen for the multiplicative noise model. Those results are due to the probability to sample $1+\mathcal{N}$ smaller than 0 and are therefore intrinsic to the noise model and not to the '+' strategy. The probability that $1+\mathcal{N}$ can be very small, in which case theory predicts divergence that will not be observed in simulations. We decided to present simple proofs relying on Borel-Cantelli Lemma. As a consequence, those proofs do not show the log-linear convergence and divergence observed in Section 2. Obtaining the log-linear behavior can be achieved using the theory of Markov chain on continuous state space. Last, we did not include results concerning a translated sphere $f(x) = \|x\|^2 + \alpha$ with $\alpha \geq 0$ for which our proofs of convergence can be extended but where linear convergence does not hold anymore due to the fact that the variance of the noise distribution does not reduce to zero close to the optimum.

Acknowledgments

The authors would like to thank Nikolaus Hansen for many valuable discussions. This work receives partial supports from the ANR/RNTL project Optimisation Multidisciplinaire (OMD).

[9] The indicator function $1_{[-a,a]}(x)$ equals 1 if $x \in [-a,a]$ and 0 otherwise.

[10] For computing the lower bound we use the fact that $P(\mathcal{N}(0,1) < x) \leq \exp(-x^2/2)/|x|\sqrt{(2\pi)}$ for $x < 0$.

References

1. Arnold, D.V.: Noisy Optimization with Evolution Strategies. GENA. Kluwer Academic Publishers, Dordrecht (2002)
2. Arnold, D.V., Beyer, H.-G.: Efficiency and mutation strength adaptation of the $(\mu/\mu_\mathrm{I}, \lambda)$-ES in a noisy environment. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 39–48. Springer, Heidelberg (2000)
3. Arnold, D.V., Beyer, H.-G.: Investigation of the (μ, λ)-ES in the presence of noise. In: Proceedings of 2001 IEEE Congress on Evolutionary Computation, pp. 332–339. IEEE Press, Los Alamitos (2001)
4. Arnold, D.V., Beyer, H.-G.: Local performance of the (1+1)-ES in a noisy environment. IEEE Transactions on Evolutionary Computation 6(1), 30–41 (2002)
5. Arnold, D.V., Beyer, H.-G.: A comparison of evolution strategies with other direct search methods in the presence of noise. Computational Optimization and Applications 24, 135–159 (2003)
6. Auger, A., Hansen, N.: Reconsidering the progress rate theory for evolution strategies in finite dimensions. In: Press, A. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 445–452 (2006)
7. Beyer, H.-G.: Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. Computer Methods in Applied Mechanics and Engineering 186(2-4), 239–267 (2000)
8. Beyer, H.-G.: The Theory of Evolution Strategies. Natural Computing Series. Springer, Heidelberg (2001)
9. Jebalia, M., Auger, A., Liardet, P.: Log-linear convergence and optimal bounds for the $(1 + 1)$-ES. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., Lutton, E. (eds.) EA 2007. LNCS, vol. 4926, pp. 207–218. Springer, Heidelberg (2008)
10. Jin, Y., Branke, J.: Evolutionary Optimization in Uncertain Environments-A Survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
11. Rechenberg, I.: Evolutionsstrategie. Friedrich Frommann Verlag (Günther Holzboog KG), Stuttgart (1973)
12. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005 (Kanpur Genetic Algorithms Laboratory, IIT Kanpur) (May 2005)

Appendix

Proof of Proposition 1. The sequence $(\mathcal{F}_s(\mathrm{X}_n))$ is decreasing and is lower bounded by 0 as $\mathcal{F}_s(\mathrm{X}_n) \geq \|\mathrm{X}_n\|^2 (1 + m_\mathcal{N}) \geq 0$. Therefore it converges to a limit $l \geq 0$. Let us show that $l = 0$. Let $\epsilon > 0$, we have to show that $\exists\, n_0 \geq 0$ such that $\mathcal{F}_s(\mathrm{X}_n) \leq \epsilon$ for $n \geq n_0$. Since the sequence $(\mathcal{F}_s(\mathrm{X}_n))$ is decreasing, we only have to show that $\exists\, n_0 \geq 0$ such that $\mathcal{F}_s(X_{n_0}) \leq \epsilon$. Let $\beta > 1$ and such that $[1+m_\mathcal{N}, \beta(1+m_\mathcal{N})[\subset \mathrm{supp}(1+\mathcal{N})$. In Lemma 2, we have defined the event $A_{n,\epsilon,\beta}$, shown that it is included in the event $\{\mathcal{F}_s(\mathrm{X}_{n+1}) \leq \epsilon\}$ and proved that the events $(A_{n,\epsilon,\beta})_n$ are independent. Moreover, $P(A_{n,\epsilon,\beta}) = P(\|\mathrm{e}_1+\sigma\mathrm{N}\|^2 \leq \frac{\epsilon}{(1+\beta)\theta^2(1+m_\mathcal{N})})P(1+\mathcal{N} \leq \beta(1+m_\mathcal{N}))$ (where θ is defined in Lemma 2) is a strictly positive constant for all n. Then $\sum_{n=0}^{+\infty} P(A_n) = +\infty$. This gives by BCL that $P(\overline{\lim}\, An) = 1$. Therefore $P(\overline{\lim}\ \{\mathcal{F}_s(\mathrm{X}_{n+1}) \leq \epsilon\}) = 1$, i.e.

$\exists n_0$ such that $\forall n \geq n_0,\ \mathcal{F}_s(\mathrm{X}_n) \leq \epsilon$. Therefore $\mathcal{F}_s(\mathrm{X}_n)$ converges to 0. The sequence $(\|\mathrm{X}_n\|)$ converges also to 0 as $\|\mathrm{X}_n\|^2 \leq \frac{\mathcal{F}_s(\mathrm{X}_n)}{1+m_{\mathcal{N}}}$. □

Lemma 2. *If $m_{\mathcal{N}} + 1 > 0$, the following points hold:*

1. *The sequence $(\|\mathrm{X}_n\|)$ is upper bounded by $\theta := \sqrt{\frac{\mathcal{F}_s(\mathrm{X}_0)}{1+m_{\mathcal{N}}}} > 0$.*
2. *Let $\epsilon > 0$ and $\beta > 1$ such that $\beta(1+m_{\mathcal{N}}) \in$ supp$(1+\mathcal{N})$. For $n \geq 0$, the event $A_{n,\epsilon,\beta} := \left(\left\{\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathrm{N}_n\right\|^2 \leq \frac{\epsilon}{(1+\beta)\theta^2(1+m_{\mathcal{N}})}\right\} \cap \{1+\mathcal{N}_n \leq \beta(1+m_{\mathcal{N}})\}\right)$[11] verifies $A_{n,\epsilon,\beta} \subset \{\mathcal{F}_s(\mathrm{X}_{n+1}) \leq \epsilon\}$. Moreover, the events $(A_{n,\epsilon,\beta})_n$ are independent.*

Proof. 1. For $n \geq 0$, $\mathcal{F}_s(\mathrm{X}_n) = \|\mathrm{X}_n\|^2(1+O_n) = \|\mathrm{X}_n\|^2\left(1+\mathcal{N}_{\phi(n)}\right)$ where $\phi(n)$ is the index of the last acceptance (obviously $\phi(n) \leq n$). Then, for $n \geq 0$ $\mathcal{F}_s(\mathrm{X}_n) \geq \|\mathrm{X}_n\|^2(1+m_{\mathcal{N}}) \geq 0$ and consequently $\|\mathrm{X}_n\|^2 \leq \frac{\mathcal{F}_s(\mathrm{X}_n)}{1+m_{\mathcal{N}}} \leq \frac{\mathcal{F}_s(\mathrm{X}_0)}{1+m_{\mathcal{N}}}$.
2. Let $\epsilon > 0$ and $\beta > 1$ such that $[1+m_{\mathcal{N}}, \beta(1+m_{\mathcal{N}})[\subset$ supp$(1+\mathcal{N})$ (with $\beta m_{\mathcal{N}} < M_{\mathcal{N}}$ if $M_{\mathcal{N}} < +\infty$). For $n \geq 0$, the event $\left\{\left(\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathrm{N}_n\right\|^2 < \frac{\epsilon}{(1+\beta)\theta^2(1+m_{\mathcal{N}})}\right) \cap (1+\mathcal{N}_n < \beta(1+m_{\mathcal{N}}))\right\}$ implies for the offspring $\tilde{\mathrm{X}}_n := \mathrm{X}_n + \sigma\|\mathrm{X}_n\|\mathrm{N}_n$ created at the iteration n that
$\mathcal{F}_s(\tilde{\mathrm{X}}_n) = \|\mathrm{X}_n\|^2\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathrm{N}_n\right\|^2(1+\mathcal{N}_n) \leq \theta^2\frac{\epsilon}{(1+\beta)(1+m_{\mathcal{N}})\theta^2}\beta(1+m_{\mathcal{N}})$.
Then $\mathcal{F}_s(\tilde{\mathrm{X}}_n) \leq \frac{\beta}{\beta+1}\epsilon < \epsilon$. If this offspring is accepted then $\mathcal{F}_s(\mathrm{X}_{n+1}) < \epsilon$, otherwise the fitness is already less than ϵ and we have also $\mathcal{F}_s(\mathrm{X}_{n+1}) < \epsilon$. Finally, the independency of the events $(A_{n,\epsilon,\beta})_n$ result from Lemma 3 applied to the sequence (X_n). □

Lemma 3. *Let (U_n) be a sequence of random vectors in $\mathbb{R}^d$ such that $P(\|U_n\| = 0) = 0$ and N_n independent random vectors distributed as $\mathrm{N}(0, I_d)$. Then the variables $Y_n := \left\|\frac{U_n}{\|U_n\|} + \sigma\mathrm{N}_n\right\|$ are independent.*

Proof. The independance of the random variables Y_n is due to the fact that the multivariate Gaussian variable $\mathrm{N}(0, I_d)$ is isotropic and is therefore invariant by rotation. The length of the vector $\frac{U_n}{\|U_n\|} + \sigma\mathrm{N}_n$ will therefore be independent of where we start on the unit hypersphere, i.e., independent of the vector $\frac{U_n}{\|U_n\|}$. □

Proof of Proposition 2. Let $n \geq n_1$ (n_1 defined in Lemma 4). We have to show that for any $m < \mathcal{F}_s(X_{n_1}) < 0$, $\exists\, n \geq n_1$ such that $\mathcal{F}_s(\mathrm{X}_n) \leq m$, or equivalently $|\mathcal{F}_s(\mathrm{X}_n)| \geq |m|$. Similarly to the proof of Proposition 1, by BCL we have ($B_{n,m,\beta}$ i.o.) (($B_{n,m,\beta}$ being defined in Lemma 4) therefore Lemma 4 gives that ($\mathcal{F}_s(\mathrm{X}_{n+1}) \leq m$ i.o.). Then $\mathcal{F}_s(\mathrm{X}_n) = \|\mathrm{X}_n\|^2(1+O_n)$ tends to $-\infty$. For all $n \geq n_1$, $0 \geq 1+O_n \geq 1+m_{\mathcal{N}}$, then $\frac{|\mathcal{F}_s(\mathrm{X}_n)|}{|1+m_{\mathcal{N}}|} \leq \|\mathrm{X}_n\|^2$ for $n \geq n_1$. Consequently $(\|\mathrm{X}_n\|)$ converges to $+\infty$ almost surely. □

Lemma 4. *Assume that $m_{\mathcal{N}} + 1 < 0$. The following points hold:*

[11] The multivariate Gaussian distribution is absolutely continuous with respect to the Lebesgue measure such that $P(\|\mathrm{X}_n\| = 0) = 0$ and then we can divide by $\|\mathrm{X}_n\|$ almost surely.

1. *There exists* $n_1 \geq 0$ *and* $A := \sqrt{\frac{|\mathcal{F}_s(X_{n_1})|}{|1+m_{\mathcal{N}}|}} > 0$ *such that* $\mathcal{F}_s(\mathrm{X}_n) < 0$ *and* $\|\mathrm{X}_n\| \geq A$ *for* $n \geq n_1$ *almost surely.*
2. *Let* $m < \mathcal{F}_s(X_{n_1}) < 0$ *and* $\beta > 1$. *For* $n \geq n_1$*, the event* $B_{n,m,\beta}$ *defined by* $B_{n,m,\beta} := \left(\left\{|1-\sigma\|\mathrm{N}_n\||^2 \geq \frac{|m|}{|m_{\mathcal{N}}+1|}\frac{\beta+1}{A^2}\right\} \cap \left\{1+\mathcal{N}_n \leq \frac{1+m_{\mathcal{N}}}{\beta}\right\}\right)$ *verifies* $B_{n,\epsilon,\beta} \subset (\mathcal{F}_s(\mathrm{X}_{n+1}) \leq m)$.

Proof. 1. We first prove that the event $\mathcal{A} := \{\, \exists\, n_1 \geq 0$ such that $\forall\, n \geq n_1$, $\mathcal{F}_s(\mathrm{X}_n) < 0\}$ is equivalent to the event $\mathcal{B} := \{\, \exists\, p_0 \geq 0$ such that $\mathcal{N}_{p_0} < -1\,\}$. Proving that $\mathcal{A} \subset \mathcal{B}$ is equivalent to show that $\mathcal{B}^c \subset \mathcal{A}^c$. Suppose that $\forall p \geq 0,\ \mathcal{N}_p \geq -1$. Then $\forall p \geq 0,\ O_p \geq -1$. Therefore $\forall p \geq 0,\ \mathcal{F}_s(X_p) = \|X_p\|^2\,(1+O_p) \geq 0$. Now we have to show that $\mathcal{B} \subset \mathcal{A}$: Suppose that $\exists\ p_0 \geq 0$ such that $\mathcal{N}_{p_0} < -1$. We denote $p_1 \geq 0$ the integer defined by $p_1 = \min\{p \in \mathbb{N}$ such that $\mathcal{N}_p < -1\}$. Then $\mathcal{F}_s\,(X_{p_1}) < 0$ and $\mathcal{F}_s\,(X_p) \geq 0$ for all $0 \leq p \leq p_1 - 1$. Since $(\mathcal{F}_s\,(\mathrm{X}_n))$ is a decreasing sequence, $\mathcal{F}_s(\mathrm{X}_n) < 0\ \forall\ n \geq p_1$. This implies that $P(\mathcal{A}) = P(\mathcal{B})$. Now, we have for all $n \geq 0$, $P(\mathcal{B}^c) = P(\cap_{p=0}^{+\infty}\,(\mathcal{N}_p \geq -1)) \leq \Pi_{p=0}^{n} P\,(\mathcal{N}_p \geq -1) = (P\,(\mathcal{N} \geq -1))^n$.

Let $a := P\,(\mathcal{N} \geq -1)$[12]. As $m_{\mathcal{N}} < -1$, then $a < 1$ which gives $P(\mathcal{B}^c) = 0$ and therefore $P(\mathcal{A}) = 1$. Then $\exists\ n_1 \geq 0$ such that $\mathcal{F}_s(\mathrm{X}_n) < 0$ for $n \geq n_1$ almost surely. The sequence $(\mathcal{F}_s(\mathrm{X}_n))_n$ is decreasing (because of the elitist selection). Then for $n \geq n_1$, $\mathcal{F}_s(\mathrm{X}_n) \leq \mathcal{F}_s(X_{n_1}) < 0$. This gives $|\mathcal{F}_s(\mathrm{X}_n)| \geq |\mathcal{F}_s(X_{n_1})| > 0$. It is easy to see (from Eq. 4) that for all $n \in \mathbb{N}$, $O_n = \mathcal{N}_{\psi(n)}$ where $\psi(n)$ is the last acceptance index before the iteration n. Combining this with the fact if $1 + m_{\mathcal{N}} \leq 1 + \mathcal{N}_{\psi(n)} < 0$ one gets $0 < |\mathcal{F}_s(X_{n_1})| \leq |\mathcal{F}_s(\mathrm{X}_n)| = \|\mathrm{X}_n\|^2|1 + \mathcal{N}_{\psi(n)}| \leq \|\mathrm{X}_n\|^2|1 + m_{\mathcal{N}}|$. Then $\|\mathrm{X}_n\|^2 \geq \frac{|\mathcal{F}_s(X_{n_1})|}{|1+m_{\mathcal{N}}|} > 0$.

2. By the first result of the Lemma, $\exists\ n_1 \geq 0,\ A > 0$ such that $\mathcal{F}_s(\mathrm{X}_n) < 0$ and $\|\mathrm{X}_n\| \geq A\ \forall n \geq n_1$. We consider $n \geq n_1$, then $\|\mathrm{X}_n\| > A$. We notice that $\forall\ y \in \mathbb{R}^d\backslash\{(0,0)\}$, $\left\|\frac{y}{\|y\|} + \sigma\mathrm{N}\right\| \geq |1 - \sigma\|\mathrm{N}\||$. Let $\beta > 1$. As the upper bound $M_{\mathcal{N}}$ verifies $1 + M_{\mathcal{N}} > 0$, $\frac{1+m_{\mathcal{N}}}{\beta} \in \mathrm{supp}(1+\mathcal{N}) \cap \mathbb{R}^-$. Suppose that we have $|1 - \sigma\|\mathrm{N}_n\||^2 \geq \frac{(\beta+1)|m|}{A^2|1+m_{\mathcal{N}}|}$ and $|1 + \mathcal{N}_n| \geq \frac{|1+m_{\mathcal{N}}|}{\beta}$, then the offspring $\tilde{\mathrm{X}}_n := \mathrm{X}_n + \sigma\|\mathrm{X}_n\|\mathrm{N}_n$ is such that $|\mathcal{F}_s(\tilde{\mathrm{X}}_n)| = \|\mathrm{X}_n\|^2\left\|\frac{\mathrm{X}_n}{\|\mathrm{X}_n\|} + \sigma\mathrm{N}_n\right\|^2|1+\mathcal{N}_n| \geq \|\mathrm{X}_n\|^2|1-\sigma\|\mathrm{N}_n\||^2|1+\mathcal{N}_n|$. Then $|\mathcal{F}_s(\tilde{\mathrm{X}}_n)| \geq \frac{\beta+1}{\beta}|m| > |m|$ which gives $\mathcal{F}_s(\mathrm{X}_{n+1}) \leq \mathcal{F}_s(\tilde{\mathrm{X}}_n) \leq m$. Consequently, for $n \geq n_0$, the event $B_{n,m,\beta} := \left\{|1 - \sigma\|\mathrm{N}_n\|^2| \geq \frac{(\beta+1)|m|}{A^2|1+m_{\mathcal{N}}|}\right\} \cap \left\{|1 + \mathcal{N}_n| \geq \frac{|1+m_{\mathcal{N}}|}{\beta}\right\}$ is included in $\{\mathcal{F}_s(\mathrm{X}_{n+1}) \leq m\}$. □

[12] We apply the same reasoning with $a = 2/3$ for the example given in the introduction where $\mathcal{N}$ take values in $\{-\gamma, 0, \gamma\}$ (with $\gamma > 1$).

Premature Convergence in Constrained Continuous Search Spaces

Oliver Kramer

Computational Intelligence Group,
Dortmund University of Technology
Otto-Hahn-Str. 14, 44227 Dortmund, Germany
oliver.kramer@tu-dortmund.de

Abstract. The optimum of numerical problems quite often lies on the constraint boundary or even in a vertex of the feasible search space. In such cases the evolutionary algorithm (EA) frequently suffers from premature convergence because of a low success probability near the constraint boundaries. We analyze premature fitness stagnation and the success rates experimentally for an EA using self-adaptive step size control. For a (1+1)-EA with a Rechenberg-like step control mechanism we prove premature step size reduction at the constraint boundary. The proof is based on a success rate analysis considering a simplified mutation distribution model. From the success rates and the possible state transitions, the expected step size change can be derived at each step. We validate the theoretical model with an experimental analysis.

Keywords: Premature Convergence, Constrained Real-Parameter Optimization, Evolution Strategies, Self-Adaptation.

1 Introduction

Whenever the search space is restricted due to constraints of the underlying problem, the EA has to make use of heuristic extensions which are called constraint handling methods. Constraint handling is very relevant to practical applications. Although the effect of premature convergence at the constraint boundary is already known, to the best of our knowledge no theoretical investigation on this topic has been published yet. Premature convergence at the constraint boundary is experimentally analyzed in section 2 and proven theoretically for simplified conditions in section 3.

1.1 Constrained Real-Parameter Optimization

A constraint is a restriction on possible value combinations of variables. In the N-dimensional search space $\mathbb{R}^N$ the task is to find an optimal solution $\boldsymbol{x}$ which minimizes $f(\boldsymbol{x})$ with subject to

$$\begin{array}{ll}\text{inequalities } g_i(\boldsymbol{x}) \leq 0, \ i = 1, \ldots, n_1, \text{ and} \\ \text{equalities} \quad h_j(\boldsymbol{x}) = 0, j = 1, \ldots, n_2.\end{array} \tag{1}$$

A feasible solution $\boldsymbol{x}$ satisfies all n_1 inequality and n_2 equality constraints.

1.2 Premature Convergence

Premature convergence of the mutation strength belongs to the most frequent problems of self-adaptive ES. As evolution rewards short term success the evolutionary process can get stuck in local optima and suffer from premature convergence. The problem is well known and experimentally proved. Only few theoretical works concentrate on this phenomenon, e.g. from Rudolph [8]. Stone and Smith [10] came to the conclusion that low mutation rates and high selection pressure result in low diversity. Another reason for premature convergence was revealed by Liang et al. [6], who point out that a solution with a high fitness but a far too small step size in one dimension is able to cause stagnation by inheriting this mutation strength to all descendants. Meyer-Nieberg and Beyer [7] point out that the reason for premature convergence could be that the operators do not fulfill the postulated requirements for mutation operators. Hansen [2] examined the conditions under which self-adaptation fails, in particular the inability of the step sizes to increase. He tries to answer the question whether a step size increase is affected by a bias of the genetic operators or due to the link between objective and strategy parameters.

2 Experimental Analysis

ES on constrained optimization problems may suffer from premature convergence in case of active inequality constraints. Broadly speaking, the reason for the premature step size reduction is the fact that the constrained region cuts off the mutative success area. Consequently, the self-adaptation process favors smaller step sizes, whose success area is not cut off.

2.1 Premature Convergence at the Constraint Boundary

The premature step size reduction has already been shown experimentally by Kramer and Schwefel [4] and for the motivation of biased mutation for ES [5]. Here, we reappraise the experiments on Schwefel's problem 2.40 [9] for the death penalty method[1] and the dynamical penalty function by Joines and Houck [3], see Table 1, in order to emphasize the success rate situation. Problem 2.40 minimizes $F(x) = -\sum_{i=1}^{5} x_i$ subject to the constraints

$$G_j(x) = \begin{cases} x_j \geq 0, & \text{for } j = 1, ..., 5 \\ -\sum\limits_{i=1}^{5}(9+i)x_i + 50000 \geq 0, & \text{for } j = 6. \end{cases} \qquad (2)$$

with minimum $x^* = (5000, 0, 0, 0, 0)^T$ and $F(x^*) = -5000$. Each row of Table 1 shows the results of a (15,100)-ES after 50 runs. As termination condition fitness stagnation is chosen: If the difference between the fitness value of the best individual of a generation and the best of the following generation is smaller

[1] Death penalty produces mutations until λ feasible exist.

Table 1. Experimental results of *death penalty* (DP) and the dynamic penalty function by Joines and Houck (Dyn) on Schwefel's problem 2.40 [9]. None of both techniques is able to approximate the optimum in any run.

	best	mean	worst	dev
DP	-4948.079	-4772.338	-4609.985	65.2
Dyn	-4780.554	-4559.129	-4358.446	85.0

than a $\theta = 10^{-12}$, then the ES terminates as the magnitude of the steps sizes is too small to effect further improvements. The ES makes use of uncorrelated mutation with N step sizes, intermediate recombination and initializes the solutions at the point $x^{(0)} = (250, 250, 250, 250, 250)^T$. Both constraint-handling methods are not able to approximate the optimum of the problem satisfactorily. The standard deviations *dev* show that the algorithms produce different results in the various runs.

2.2 Success Rate Analysis

An experimental analysis of the success rates at the constraint boundary of 2.40 help to understand the situation. The success rate p_s is the ratio of successful, i.e. feasible *and* better mutations $\mathcal{S}$, and all mutations $\mathcal{A}$: $(p_s) = \frac{|\mathcal{S}|}{|\mathcal{A}|}$. Figure 1 shows the success probabilities to produce *feasible* offspring (1), to produce *better* offspring (2) and to produce *feasible better* (3) offspring, i.e. (p_s), in the neighborhood of the constraint boundary. On the feasible surface of a hypersphere with radius r around the optimum $x^{(*)}$, 1000 points have been generated. Around every generated point 1000 samples were generated on the surface of a hypersphere with radius $\hat{\sigma} \cdot r$. The success rate situation can be generalized because of the linearity of the fitness and the constraint functions: it depends on the ratio $r/\hat{\sigma}$. The probability for feasibility is almost one for $\hat{\sigma} < 0.001$ and decreases to zero between $0.001 < \hat{\sigma} < 1$. Consequently, the success rate (p_s) decreases within the same interval.

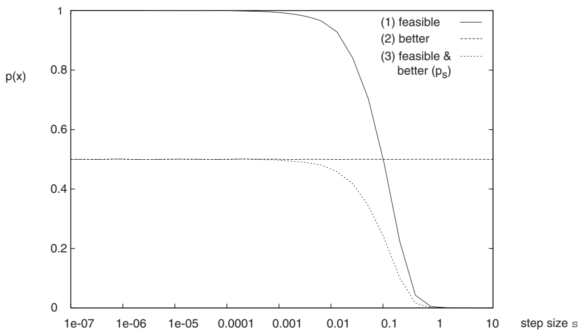


Fig. 1. The success probabilities p_s in the neighborhood of the constraint boundary on Schwefel's problem 2.40

From this success probability analysis and experimentally determined successful μ/λ-rates we try to ascertain an optimal normalized step size $\hat{\sigma}$. An increase of the number of offspring yields the rate $\frac{\mu}{\lambda} = \frac{15}{700}$, i.e. from $\lambda > 700$ premature convergence can be prevented with $\mu = 15$ on problem 2.40 in every run. A (15,700)-ES achieves the approximation with an normalized[2] step size of $\hat{\sigma} \approx 0.12$. This observation corresponds to the success rate analysis of plot 1. The success rate of $(p_s) = \frac{15}{700} \approx 0.021$ yields a normalized step size between 0.1 and 0.5, which matches the observation $\hat{\sigma} \approx 0.12$. Interestingly, this resembles to the optimal normalized step size $\hat{\sigma} \approx 0.12$ which could be shown for the sphere function [1], but with a success rate of $(p_s) \approx 0.27$.

3 Theoretical Analysis

In this section we analyze the premature convergence at the constraint boundary with a simple (1+1)-EA using a Rechenberg-like step size adaptation method. From an analysis of the success rates with linear constraints and a linear fitness function we can describe the behavior of a Markov model of the (1+1)-EA controlling the step sizes. After the proof of premature convergence we validate the model experimentally using Gaussian mutation.

3.1 A Proof of Premature Convergence for a (1+1)-EA

We consider the two dimensional case of a linear objective function and one linear constraint with the angle β between the latter and the contour lines of the fitness function at the location where the EA first encounters the constraint boundary. We assume that the optimum is not located at the same position and that the EA has to move along the constraint boundary to converge. Figure 2 shows the success rate situations of individual $\boldsymbol{x}$ with distance d to the constraint boundary for the three cases: 1. $\sigma < d$, i.e. not constrained, and constrained 2. $\sigma > d$, $\sigma < s$ and 3. $\sigma > d$, $\sigma > s$. We analyze the behavior of a (1+1)-EA with adaptive step sizes modeled[3] by the Markovian process $(X_t, \sigma_t)_{t \geq 0}$ generated by

$$X_{t+1} = \begin{cases} X_t + \sigma_t Z_t & \text{if } f(X_t + \sigma_t Z_t) < f(X_t) \\ & \quad \wedge g(X_t + \sigma_t Z_t) = 0 \\ X_t & \text{otherwise} \end{cases} \tag{3}$$

and

$$\sigma_{t+1} = \begin{cases} \gamma \sigma_t & \text{if } f(X_t + \sigma_t Z_t) < f(X_t) \\ & \quad \wedge g(X_t + \sigma_t Z_t) = 0 \\ \gamma^{-1} \sigma_t & \text{otherwise} \end{cases} \tag{4}$$

with step size σ_t and mutation parameter $\gamma > 1$. The function g measures the constraint violation. Each random vector $Z_t, t \geq 0$ is independent and identically

[2] Ratio between average step sizes of μ parents and the distance to optimum.

[3] As Rudolph [8] states, this EA does not exactly match a (1+1)-EA with self-adaptive step size control, but it can be transferred to a broader class of EAs.

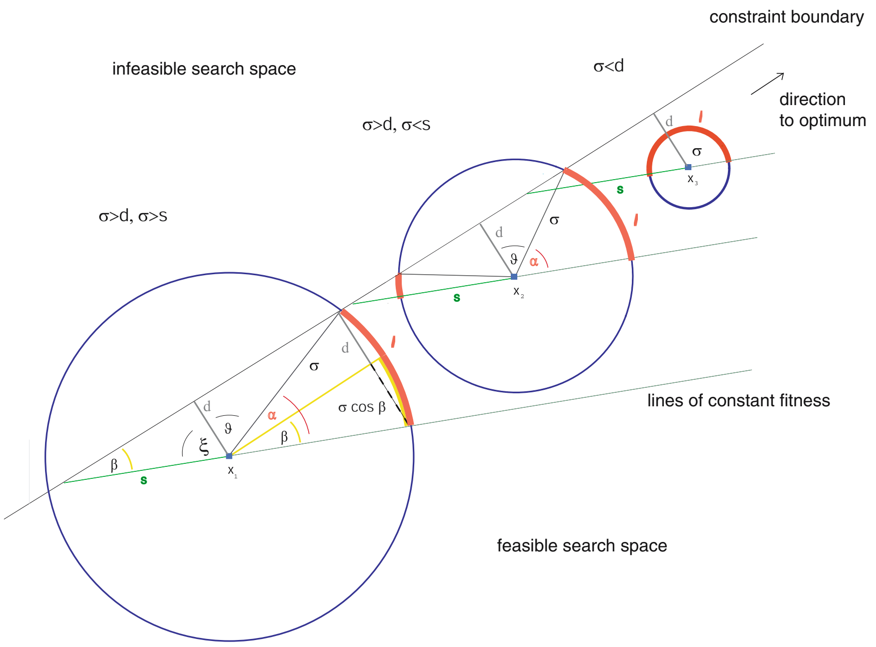


Fig. 2. Success rates at the boundary of the feasible search space. Three cases have to be considered, i.e. 1. $\sigma < d$, 2. $\sigma > d$, $\sigma < s$ and 3. $\sigma > d$, $\sigma > s$. The bold circular arcs are the regions where successful mutations are produced.

distributed in the following way: We assume that mutations $\sigma_t Z_t$ are produced on the edge of the circle around X_t with radius σ_t. When a successful mutation is produced, the step length σ_t is increased and decreased otherwise. We are interested in the development of the step size σ_t and the distance d_t to the constraint boundary. For the sake of better readability we write σ instead of σ_t and d instead of d_t where possible. In the following lemma 1 we analyze the success probabilities for the three cases.

Lemma 1. *Let (p_s) be the success probability for individual X_t of the (1+1)-EA, with step size σ and distance d to the constraint boundary. Then it holds $(p_s)_{\sigma<d} = 1/2$ and $(p_s)_{\sigma>d} < 1/2$. For $d/\sigma \rightarrow 0$ it holds $(p_s)_{\sigma>d} \rightarrow \beta/(2\pi)$.*

Proof. The analysis of the probabilities for changing through the states is based on a success rates analysis of the circle model for the three mentioned cases, see figure 2. In our model the success rate p_s is the relation between the length of the circular arc $l = 2\pi r\alpha/(2\pi)$ and the circumference $2\pi r$:

$$(p_s) = \alpha/(2\pi). \tag{5}$$

The EA starts its run in the feasible part of the search space, $\sigma < d$. As the fitness function is linear and the constraint boundary does not cut off the circle, the angle over the circular arc with successful mutations is $\alpha = \pi$. Hence,

$$\boxed{(p_s)_{\sigma<d} = 1/2} \tag{6}$$

The step sizes are increased with probability 1/2 and decreased with probability 1/2. For $\sigma > d$ we have to distinguish two cases:

1. $\sigma < s$, i.e. the circle cuts its contour line within the feasible region. Distance s is the segment of the contour line between x and the constraint boundary with $s = (d/\sin\beta)$. The success area on the edge of the circle is the circular arc over α and the circular arc in the opposite direction. The success probability $(p_s)_{\sigma>d,\sigma<s}$ can also be expressed with the help of ϑ

$$(p_s)_{\sigma>d,\sigma<s} = 1/2 - \vartheta/\pi \tag{7}$$

The triangle with a right angle yields $\cos(\vartheta) = d/\sigma$ and the success probability becomes

$$(p_s)_{\sigma>d,\sigma<s} = 1/2 - \arccos(d/\sigma)/\pi \tag{8}$$

As $0 < (d/\sigma) < 1$, it holds $0 < \arccos(d/\sigma) < \pi/2$ and the desired property

$$\boxed{(p_s)_{\sigma>d,\sigma<s} < 1/2} \tag{9}$$

The success probability $(p_s)_{\sigma>d,\sigma<s}$ is comparatively small. Analyzing the asymptotic behavior we get

$$(p_s)_{\sigma>d,\sigma<s} \to 0 \text{ for } d/\sigma \to 0, \tag{10}$$

i.e. for small distances $d \to 0$ or huge step sizes $\sigma \to \infty$ the success probability becomes 0. But one have to keep in mind that a small d/σ implies a small angle β as $\sigma < s$. It is obvious that $(p_s)_{\sigma>d,\sigma<s} \geq (p_s)_{\sigma>d,\sigma>s}$, hence the analysis of the case $\sigma > s$ is more interesting as it considers β.

2. $\sigma > s$, i.e. the circle cuts its contour line beyond the constraint boundary. The angle α is $\pi - (\xi + \vartheta)$. In the triangle of β and ξ it holds $\xi = \pi/2 - \beta$, we get $\alpha = \beta + \pi/2 - \arccos(d/\sigma)$. Hence, the success probability is

$$(p_s)_{\sigma>d,\sigma>s} = \alpha/(2\pi) = \beta/(2\pi) + (\pi/2 - \arccos(d/\sigma))/(2\pi) \tag{11}$$

Again, for $0 < d/\sigma < 1$, it holds $0 < \arccos(d/\sigma) < \pi/2$ and as we postulated $\beta = \pi/2$, we get

$$\boxed{(p_s)_{\sigma>d,\sigma>s} < 1/2} \tag{12}$$

As $(p_s)_{\sigma>d,\sigma<s} \geq (p_s)_{\sigma>d,\sigma>s}$ for a small distance d or a big step size σ, we can assert the asymptotic behavior

$$(p_s)_{\sigma>d} \to \beta/(2\pi) \text{ for } d/\sigma \to 0 \tag{13}$$

Furthermore, $(p_s)_{\sigma>d}$ decreases for smaller β. □

To illustrate the success rate situation, we assume $\beta = \pi/4$ and $d/\sigma = 1/2$. As $2d > (d/\sin\beta)$, we have to consider the case $\sigma > s$. The success rate becomes $(p_s)_{\sigma>d,\sigma>s} = 5/24$. As the probability for a small step size $(p_s)_{\sigma<d} = 1/2$ is much bigger, it will in particular be preferred by a self-adaptive mechanism.

We continue to prove the step size reduction for our (1+1)-EA. As the optimum lies at the constraint boundary, the EA will *encounter* the latter, $\sigma > d$. Theorem 1 describes the behavior of the (1+1)-EA at the constraint boundary by stating the expected mutation change $E(\gamma_{t+1})$ in each iteration, lemma 1 provides a quantitative analysis stating the success rate $(p_s)_{\sigma>d}$, in the following denoted as p.

Theorem 1 (Premature Step Size Reduction). *Let f be a linear fitness function and g a linear constraint boundary with angle $\beta < \pi/2$ between f and g. In the vicinity of the constraint boundary $\sigma > d$, the above modeled (1+1)-EA with $\gamma > 2$ reduces its step size σ in each iteration by $E(\gamma_{t+1}) = \gamma_t^{\frac{1}{(1-p)}-\frac{1}{p}} < 1$ with $p < 1/2$.*

Proof. We distinguish two states Γ_1 and Γ_2 for an individual X_t in the neighborhood of the constraint boundary, $\sigma > d$. State Γ_1 denotes the situation of **success**, i.e. $f(X_t + \sigma_t Z_t) < f(X_t) \wedge g(X_t + \sigma_t Z_t) = 0$. Γ_2 denotes the **failure** $f(X_t + \sigma_t Z_t) > f(X_t) \vee g(X_t + \sigma_t Z_t) > 0$. Lemma 1 shows that the probability for a success is small, at most smaller than 1/2 and converges to $\beta/(2\pi)$ for $d/\sigma \to 0$. The state transitions between the two states Γ_1 and Γ_2 are analyzed in the following.

- $\Gamma_1 \to \Gamma_1$. The probability for a success is p, which is relatively low, see lemma 1. A success results in a step size increase $\sigma_{t+1} = \gamma\sigma_t$. A successful mutation may lie arbitrarily close to the constraint boundary ($d/\sigma \to 0$) and therefore decrease the success probability p rapidly. But a success may also lead to an increase of the distance to the constraint boundary, at most by $\sigma \sin\beta$ (dotted line). The constrained case is not left, if the step size increase is higher than the distance increase to guarantee $d/\sigma < 1$:

$$\frac{d + \sigma \sin\beta}{\sigma\gamma} < 1 \equiv 2 < \gamma \tag{14}$$

  As $d/\sigma < 1$ it must hold $\gamma > 2$ to fulfill the above condition for the proof of step size reduction. So, the probability for staying in state Γ_1 is

$$P(\Gamma_1 \to \Gamma_1 | \Gamma_1) = p, \quad \sigma_{t+1} = \gamma\sigma_t. \tag{15}$$

- $\Gamma_2 \to \Gamma_1$: The probability for a success if the last step was a failure is again p, the step size is increased and the constrained case is not left for $\gamma > 1 + \tan\beta$.

$$P(\Gamma_2 \to \Gamma_1 | \Gamma_2) = p, \quad \sigma_{t+1} = \gamma\sigma_t. \tag{16}$$

- $\Gamma_1 \to \Gamma_2$: The probability for a failure is $1 - p$. It results in a step decrease $\sigma_{t+1} = \gamma^{-1}\sigma_t$. If the step decrease leads to $d/\sigma_{t+1} > 1$, the constraint boundary is left. In this case step size decrease and increase occur with the same probability and the expected change of σ becomes $E(\gamma_{t+1}) = \gamma_t \cdot \gamma_t^{-1} = 1$. But the constraint boundary will be reached again within the following steps. Hence, we summarize

$$P(\Gamma_1 \to \Gamma_2|\Gamma_1) = 1 - p, \quad \sigma_{t+1} = \gamma^{-1}\sigma_t. \tag{17}$$

– $\Gamma_2 \to \Gamma_2$: Similar to transition $\Gamma_1 \to \Gamma_2$ the probability to stay in the state Γ_2 is the probability $1 - p$ for a failure.

$$P(\Gamma_2 \to \Gamma_2|\Gamma_2) = 1 - p, \quad \sigma_{t+1} = \gamma^{-1}\sigma_t. \tag{18}$$

This yields the following state transition probability matrix $\boldsymbol{T}$ for states Γ_1 and Γ_2:

$$\boldsymbol{T} = \begin{pmatrix} p & (1-p) \\ p & (1-p) \end{pmatrix} \tag{19}$$

It is worth to mention that in the case of a success with an overwhelming probability of $p' = (1 - \beta/(2\pi)) \to 1$ for $\beta \to 0$, the distance d to the constraint boundary is decreased. This condition also contributes to an iterative reduction of the distance to the constraint boundary before reaching the optimum. From the probability of each state transition and the step size change, an expected step size change can be derived. Figure 3 shows the probabilities for the state transitions of Γ_1 and Γ_2, together with the change of step size σ for each transition. In each state, there are only two possibilities: staying in the same state or

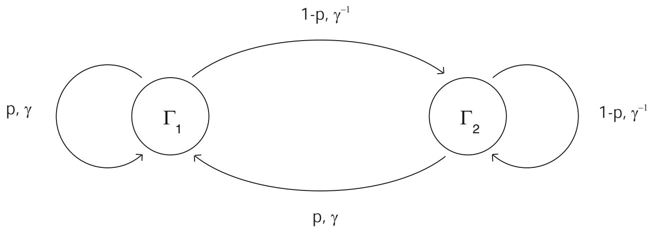


Fig. 3. The state transitions of Γ_1 and Γ_2, its probabilities at the constraint boundary and the influence on the step size σ

leaving the state. Hence, the state transitions are geometrically distributed. We are able to determine the *expected* change of the step sizes. The probability to *leave* state Γ_1 is $1-p$, so the expected number of iterations to stay is $1/(1-p)$. Each iteration the step size is increased by γ. When leaving to state Γ_2 the step size is decreased by γ^{-1}. The probability to *leave* state Γ_2 is p, so the expected number of iterations to stay is $1/p$ with a step decrease by γ^{-1}. Returning to state Γ_1 leads to a step increase by γ. From these considerations we can now determine the expected development of γ by

$$E(\gamma_{t+1}) = \gamma_t^{\frac{1}{1-p}} \cdot \gamma_t^{-1} \cdot \gamma_t^{-\frac{1}{p}} \cdot \gamma_t = \gamma_t^{\frac{1}{1-p} - \frac{1}{p}}. \tag{20}$$

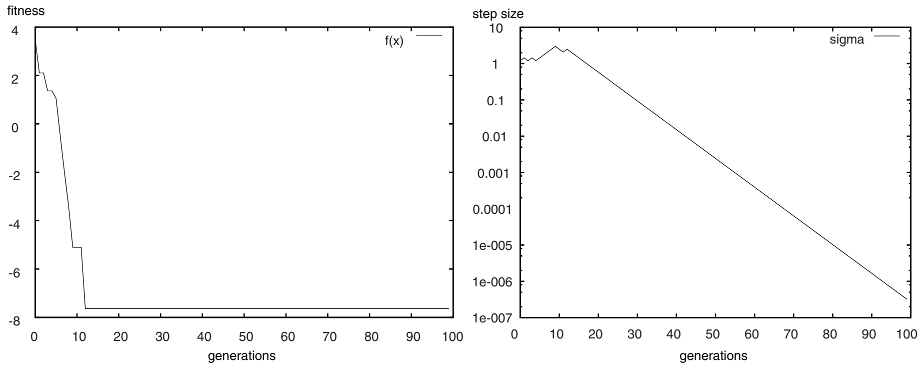


Fig. 4. Experimental validation for the step size reduction of the (1+1)-EA at the constraint boundary. The left plot shows the fitness development, the right plot confirms the reduction of σ. In generation $t \approx 12$ the EA reaches the constraint boundary.

Lemma 1 has proven that $p < 1/2$, so $\frac{1}{1-p} - \frac{1}{p} < 0$ and

$$\boxed{E(\gamma_{t+1}) = \gamma_t^{\frac{1}{1-p} - \frac{1}{p}} < 1} \tag{21}$$

The expected change of step size σ is $E(\gamma_{t+1}) < 1$ in each generation, so the steps are decreasing at the constraint boundary. For small success probabilities, see lemma 1, the decrease becomes quite high. □

3.2 Experimental Model Validation

In order to verify the theoretical model, we implemented the (1+1)-EA specified by equations 3 and 4, but based on Gaussian mutation with one step size σ. We tested various settings for γ and initializations for σ to minimize the linear function

$$f(\boldsymbol{x}) = -x_2 \quad \text{with} \quad g(\boldsymbol{x}) = -x_1 + x_2 < 0, \tag{22}$$

starting from point $x^{(0)} = (5, -5)^T$. For $N = 2$, $\sigma^{(0)} = 1$ and $\gamma = 1.2$ we observed the fitness stagnation in every of 100 runs. The stagnation could also be observed with other parameterizations. The EA is not able to move along the constraint boundary while constantly minimizing $f(x)$. Instead, it decreases the step size when reaching the vicinity of the constraint boundary – like the theoretical model predicts. Figure 4 shows the fitness development and the σ-reduction of a typical run for 100 generations. We observed the same behavior of the EA for higher dimensions.

4 Summary and Outlook

For the mutation strength control of ES, the premature fitness stagnation could be shown experimentally. We proved step size reduction for a (1+1)-EA under simplified conditions. The proof is based on a success rate analysis considering a simplified EA model on linear functions. The situation at the constraint boundary can be modeled by two different states. From the success rates and the possible state transitions, the expected step size changes in every step can be derived. We validated the (1+1)-model with a Rechenberg-like step size rule experimentally for the postulated linear conditions. From the above analysis we must conclude that we have to control the mutation rates with care and sense for the success probabilites at the constraint boundary. In the future we will try to extend the proof to more than two dimensions and for other mutation distribution functions. The model may be feasible for $p < 1/2$ which surely holds for other symmetric distributions, because a part is cut off by the constraint boundary. The question is whether the state transitions for the Markovian model can also be guaranteed for $N > 2$. We also plan to extend the argumentation to nonlinear constraints and nonlinear fitness landscapes.

References

1. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies - A Comprehensive Introduction. Natural Computing 1, 3–52 (2002)
2. Hansen, N.: An Analysis of Mutative Sigma Self-Adaptation on Linear Fitness Functions. Evolutionary Computation 14(3), 255–275 (2006)
3. Joines, J., Houck, C.: On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GAs. In: Fogel, D.B. (ed.) Proceedings of the Conference on Evolutionary Computation, pp. 579–584. IEEE Press, Orlando (1994)
4. Kramer, O., Schwefel, H.-P.: On Three New Approaches to Handle Constraints Within Evolution Strategies. Natural Computing 5(4), 363–385 (2006)
5. Kramer, O., Ting, C.-K., Büning, H.K.: A New Mutation Operator for Evolution Strategies for Constrained Problems. In: Proceedings of the Congress on Evolutionary Computation - CEC 2005, pp. 2600–2606 (2005)
6. Liang, K.-H., Yao, X., Liu, Y., Newton, C.S., Hoffman, D.: An Experimental Investigation of Self-Adaptation in Evolutionary Programming. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 291–300. Springer, Heidelberg (1998)
7. Meyer-Nieberg, S., Beyer, H.-G.: Self-Adaptation in Evolutionary Algorithms. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) Parameter Setting in Evolutionary Algorithms. Springer, Berlin (2007)
8. Rudolph, G.: Self-Adaptive Mutations Lead to Premature Convergence. IEEE Transactions on Evolutionary Computation 5(4), 410–414 (2001)
9. Schwefel, H.-P.: Evolution and Optimum Seeking. In: Sixth-Generation Computer Technology. Wiley Interscience, New York (1995)
10. Stone, C., Smith, J.: Strategy Parameter Variety in Self-Adaptation of Mutation Rates. In: Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2002, pp. 586–593. Morgan Kaufmann Publishers, San Francisco (2002)

Approximating Minimum Multicuts by Evolutionary Multi-objective Algorithms*

Frank Neumann[1] and Joachim Reichel[2]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
firstname.lastname@mpi-inf.mpg.de
[2] Institut für Mathematik, TU Berlin, Germany
reichel@math.tu-berlin.de

Abstract. It has been shown that simple evolutionary algorithms are able to solve the minimum cut problem in expected polynomial time when using a multi-objective model of the problem. In this paper, we generalize these ideas to the NP-hard minimum multicut problem. Given a set of k terminal pairs, we prove that evolutionary algorithms in combination with a multi-objective model of the problem are able to obtain a k-approximation for this problem in expected polynomial time.

1 Introduction

Evolutionary algorithms and other kinds of metaheuristics have become very popular for solving combinatorial optimization problems. In recent years, a lot of progress has been made in understanding this kind of algorithms with respect to their runtime behavior. Most of these results are on classical polynomially solvable problems such as minimum spanning trees [19] or shortest paths [9,20]. One goal of such studies on easy problems is to get an understanding how the heuristics work in order to analyze difficult problems in the future. Later on, such studies have served as a basis for analyzing evolutionary algorithms on NP-hard problems [10,16,22].

Recently, it has been shown in [17] that the minimum cut problem cannot be solved by simple single-objective evolutionary algorithms. In contrast to this a multi-objective approach has been presented which provably solves the problem in expected polynomial time. The algorithms analyzed in this paper use the dual problem, i. e., the maximum flow problem, as a subroutine. The goal of this study was to gain new insights into the behavior of evolutionary algorithms when considering cutting problems and to express the usefulness of the original problem and its dualization with respect to the optimization by evolutionary algorithms. The study carried out in the present paper extends the mentioned results to the NP-hard minimum multicut problem. In this problem a set of k pairs of nodes (s_i, t_i), $1 \leq i \leq k$ is given and the goal is to find a cut of minimum cost such that all (s_i, t_i) pairs are separated. This problem has been shown to

* This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

be MAX SNP-hard [3,7,8,21]. As a consequence, there is no polynomial time approximation scheme (unless P = NP) [2].

Due to the results obtained in [17], we consider multi-objective models for the multicut problem using flow computations which can be carried out in polynomial time as a subroutine. It is our aim to examine how such an approach can approximate an optimal solution for this problem. We study an evolutionary algorithm called Global SEMO (GSEMO) which has been widely used for the runtime analysis of evolutionary multi-objective algorithms (see e.g. [4,10,17,18]). Our analysis points out that this algorithm achieves a factor k-approximation in polynomial time as long as the weights on the edges of the given graph are polynomially bounded in the size of the input. The requirement on the edge weights is necessary since the population size of GSEMO may become as large as a polynomial in the largest edge weight [13]. One way to deal with this circumstance is to incorporate the concept of ε-dominance [14] into the algorithm. Using this mechanism in a similar way as done in [13,17], we prove that a k-approximation can be achieved in expected polynomial runtime even if the weights of the given graph are not polynomially bounded.

The paper is organized as follows. In Section 2, we present the model of the multicut problem and the algorithms that are analyzed in this paper. The results for GSEMO are presented in Section 3. In Section 4 we improve these results by incorporating the ε-dominance approach into the algorithm. Finally, we give some concluding remarks.

2 Problem Definition

We consider the following problem. Given a connected directed or undirected graph G = (V, E) on n vertices and m edges and a cost function $c : E \mapsto \mathbb{N}_+$ that imposes positive integer weights on the edges. Let $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ be a set of k pairs with $s_i \neq t_i, 1 \leq i \leq k$. The source of commodity i is given by s_i, the target by t_i. We denote by $c_{\max} = \max_{e \in E} c(e)$ the largest cost among all edges.

A multicut $S \subseteq E$ is a set of edges such that there is no path from s_i to t_i in $(V, E \setminus S)$ for any commodity i. The cost of a subset of E is defined as the sum of the costs of its elements. The goal is to find a multicut $S \subseteq E$ of minimum cost. For $k = 1$, we obtain the minimum s-t-cut problem as a special case.

The dual of this problem is the maximum-value multicommodity flow problem. This problem asks for an s_i-t_i-flow for each commodity i such that the sum of all flow values is maximum. The flow for each commodity i has to satisfy the flow conservation constraints at every node except s_i and t_i, and the sum of all k flows has to obey the capacities given by the cost function c.

Let F_i denote the value of a maximum s_i-t_i-flow in G and define $F := \sum_i F_i$. Let F^* denote the sum of all flow values of a maximum multicommodity flow in G and let C^* denote the cost of a minimum multicut of G. Note that $F^* \leq C^* \leq C := m \cdot c_{\max}$. Furthermore we have $F^* \leq F = \sum_i F_i \leq k \cdot F^* \leq k \cdot C^* \leq k \cdot C$.

For the undirected case it has been shown [11] that $C^* \in O(\log(k) \cdot F^*)$. The proof is constructive and leads to an $O(\log k)$-approximation algorithm for the minimum multicut problem. This bound is tight, i. e., there are graph classes for which $C^* \in \Omega(\log(k) \cdot F^*)$ holds. In the directed case, the gap between F^* and C^* can be as large as $\tilde{\Omega}(n^{1/7})$ [6] and is at most $O(\sqrt{n \log(k+1)})$ [5]. In the special case $k = 1$ we have $F^* = C^*$ by the max-flow-min-cut theorem [1].

Based on the results in [17], we consider an edge-based approach. We work with bit strings of length $m = |E|$. For a search point $x \in \{0,1\}^m$, the set $E(x) := \{e_i \in E \mid x_i = 1\}$ denotes the subset of E corresponding to the 1's in x. Note, that not every search point represents a multicut, i. e., not every search point is a feasible solution.

Due to the results for the special case of the minimum s-t-cut problem [17], we do not consider single-objective evolutionary algorithms at all. Instead we focus on multi-objective evolutionary algorithms. Examples for simple multi-objective evolutionary algorithms that have been analyzed before are SEMO and GSEMO [12,15,18]. The GSEMO algorithm can be described as follows. Note that the fitness function f is vector-valued and the $\leq$-comparison is to be understood component-wise.

Algorithm 1. *GSEMO (Global Simple Evolutionary Multi-objective Optimizer)*

```
1. Choose $x \in \{0,1\}^m$ uniformly at random.
2. Determine $f(x)$ and initialize $P := \{x\}$.
3. Repeat
   – choose $x \in P$ uniformly at random.
   – create an offspring $y$ by flipping each bit of $x$ independently with probability $1/m$.
   – let $P$ unchanged, if there is an $z \in P$ such that $f(z) \leq f(y)$ and $f(z) \neq f(y)$.
   – otherwise, exclude all $z$ with $f(y) \leq f(z)$ and add $y$ to $P$.
```

We consider the fitness function $f : \{0,1\}^m \mapsto \mathbb{N}^2$, $f(x) = (cost(x), flow)$, where $cost(x) = \sum_{e \in E(x)} c(e)$, $flow(x) := \sum_i flow_i(x)$ and $flow_i(x)$ denotes the value of a maximum s_i-t_i-flow in $G(x) := (V, E \setminus E(x))$.

Note that the values of both components $cost(\cdot)$ and $flow(\cdot)$ of the fitness function can be exponential in the input size, which implies that GSEMO has to cope with a Pareto front of exponential size.

Therefore, we also investigate a variation of GSEMO which uses the concept of ε-dominance [14]. It has already been shown in [13,17] that such an approach may be provably helpful when dealing with exponentially large Pareto fronts. We consider the DEMO algorithm (Diversity Evolutionary Multi-objective Optimizer) which differs from GSEMO by using a function b that assigns the same function value to search points with similar objective vectors. During the run of the algorithm at most one search point for any fixed function value of b is present in the population.

We examine DEMO partitioning the objective space into boxes by using the function $b : \{0,1\}^m \mapsto \mathbb{N}^2$ with $b_1(x) := \left\lfloor \frac{\log(1+cost(x))}{\log(1+\varepsilon)} \right\rfloor$ and $b_2(x) := \left\lfloor \frac{\log(1+flow(x))}{\log(1+\varepsilon)} \right\rfloor$, where $\varepsilon > 0$ is a parameter that determines the size of the boxes. The algorithm has the following description.

Algorithm 2. *DEMO (Diversity Evolutionary Multi-objective Optimizer)*

```
1. Choose $x \in \{0,1\}^m$ uniformly at random.
2. Determine $f(x)$ and initialize $P := \{x\}$.
3. Repeat
   – choose $x \in P$ uniformly at random.
   – create an offspring $y$ by flipping each bit of $x$ independently with probability $1/m$.
   – let $P$ unchanged, if there is an $z \in P$ such that $b(z) \leq b(y)$ and $(b(z) \neq b(y)$ or $cost(z) + flow(z) < cost(y) + flow(y))$.
   – otherwise, exclude all $z$ with $b(y) \leq b(z)$ and add $y$ to $P$.
```

The DEMO algorithm discards a new search point y if the corresponding box $b(y)$ is dominated by the box $b(z)$ of some search point $z \in P$ (and y and z do not fall into the same box). If $b(y) = b(z)$, the algorithm discards y if its sum of cost and flow value is larger than that of z. Otherwise, all search points in dominated boxes are removed from the population and y is included into the population.

Due to Laumanns et al. [14], the following upper bound on the population size can be given.

Lemma 1. *The population size $|P|$ of DEMO is upper bounded by*

$$B := \frac{\log(1+C)}{\log(1+\varepsilon)} = O(\varepsilon^{-1} \log C) = O(\varepsilon^{-1}(\log n + \log c_{\max})).$$

The algorithms described in this section do not use any stopping criteria. For theoretical investigations it is common to consider the algorithms as infinite stochastic processes and to use the number of fitness evaluations as a measure of the runtime. Our goal is to bound the expected number of fitness evaluations (also called expected runtime) until the algorithms have obtained a good approximation for the multicut problem.

We point out that we use the sum of single-commodity flow values instead of the value of a multi-commodity flow as second component of the fitness function. While the multi-commodity flow value would probably lead to a stronger approximation bound, the computation of a multi-commodity flow requires linear programming as there is no combinatorial algorithm known. On the other hand, single-commodity flows can be efficiently computed using different well-studied algorithms [1]. Furthermore, an oracle for the multi-commodity flow value is a rather strong oracle as it provides the value of the dual problem.

3 Analysis of GSEMO

The goal of this section is to prove a pseudopolynomial upper bound on the runtime of GSEMO until it has achieved an F/C^*-approximation for the multicut problem. Note that $F/C^* \leq k$, hence in the worst case we get a k-approximation. We denote by $L = \{x \in \{0,1\}^m \mid cost(x) + flow(x) \leq F\}$ the set of search points

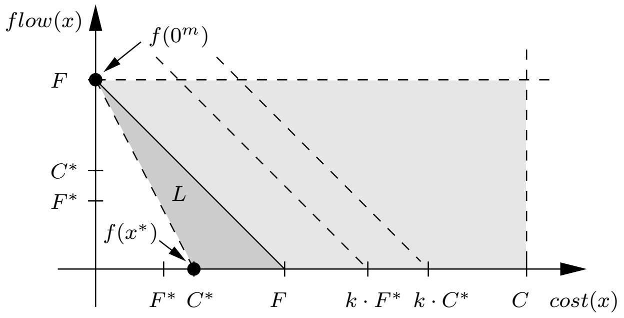


Fig. 1. Objective space of the fitness function $f(x) = (cost(x), flow(x))$. The sketch depicts the case that the sequence F^*, C^*, F, $k \cdot F^*$, $k \cdot C^*$ is strictly increasing. Note that subsequent values may coincide and that C can be as small as C^*. Optimal multicuts x^* have objective vector $(C^*, 0)$, k-approximations lie on the the segment from $(C^*, 0)$ to $(\min\{k \cdot C^*, C\}, 0)$.

whose objective vectors lie on or below the line given by the two objective values $(0, F)$ and $(F, 0)$. Figure 1 shows a graphical representation of the objective space. The following proposition shows that the search points of L represent subsets of F/C^*-approximations of minimum multicuts.

Proposition 1. *Let $x \in L$. Then $E(x)$ is a subset of an F/C^*-approximation of a minimum multicut of G.*

Proof. Since $x \in L$ we have $cost(x) + flow(x) \leq F$. Let S denote a minimum multicut of $G(x)$. Then $E(x) \dot\cup S$ is a multicut of G with $cost(E(x) \dot\cup S) = cost(x) + cost(S)$. Since S is a minimum multicut of $G(x)$, its cost is not larger than the sum of the cost of the individual minimum s_i-t_i-cuts, i. e., $cost(S) \leq flow(x)$. Hence, we have $cost(E(x) \dot\cup S) \leq cost(x) + flow(x) \leq F \leq k \cdot F^* \leq k \cdot C^*$, which implies that $E(x) \dot\cup S$ is an F/C^*-approximation of a minimum multicut of G. □

The preceding proposition implies the following condition for F/C^*-approximate solutions which will be essential for the analysis of the algorithms.

Corollary 1. *Let $x \in \{0, 1\}^m$ such that $flow(x) = 0$. Then $E(x)$ is an F/C^*-approximation of a minimum multicut of G if and only if $x \in L$.*

We remark that the converse of Proposition 1 is not true in general, in contrast to the single-commodity case $k = 1$.

For $x \in \{0, 1\}^m$ and $e \in E$ define $x^{+e} \in \{0, 1\}^m$ by $x^{+e}(e) = 1$ and $x^{+e}(e') = x(e')$ for $e' \neq e$. We can bound $flow(x^{+e})$ in terms of $flow(x)$ as follows.

Proposition 2. *Let $x \in \{0, 1\}^m$ and $e \in E$. Then $flow(x^{+e}) \geq flow(x) - kc(e)$.*

Proof. By the (single-commodity) max-flow min-cut theorem we have $flow_i(x^{+e}) \geq flow_i(x) - c(e)$ for each commodity i. Summation over i yields the claimed result. □

Proposition 3. *Let $x \in \{0,1\}^m$ such that $flow_i(x) > 0$ for some commodity i. Let $e \in E \setminus E(x)$ an edge of a minimum s_i-t_i-cut of $G(x)$. Then $flow(x^{+e}) \leq flow(x) - c(e)$ and $cost(x^{+e}) + flow(x^{+e}) \leq cost(x) + flow(x)$.*

Proof. Since $flow_i(x) > 0$ the minimum s_i-t_i-cut of $G(x)$ is not the empty set. Let $x \in E \setminus E(x)$ an edge from such a minimum s_i-t_i-cut. By the (single-commodity) max-flow min-cut theorem we have $flow_i(x^{+e}) = flow_i(x) - c(e)$. Furthermore, $flow_j(x^{+e}) \leq flow_j(x)$ holds for $j \neq i$. Summation over i yields the first claim.

Since $cost(x^{+e}) = cost(x) + c(e)$, the second claim follows directly from the first one. □

The following corollary is an immediate consequence of the preceding proposition and the definition of L.

Corollary 2. *Let $x \in L$ a search point such that $flow(x) > 0$. Then there exists a 1-bit flip leading to a search point $x' \in L$ with $flow(x') < flow(x)$.*

Now we are able to prove the following theorem which shows that the expected runtime of GSEMO is pseudopolynomial with respect to the given input.

Theorem 1. *The expected time until GSEMO working on the fitness function f constructs an F/C^*-approximation of a minimum k-commodity multicut is $O(F(\log n + \log c_{\max}))$.*

Proof. The size of the population P is at most F as GSEMO keeps at each time at most one solution per fixed flow value. First we consider the time until $0^m \in L$ has been included into the population. Note that $cost(0^m) = 0$. Afterwards we study the time until $x \in L$ with $flow(x) = 0$ has been included. By Corollary 1 the edge set $E(x)$ is an F/C^*-approximation of a minimum multicut.

The expected time until GSEMO working on the fitness function f constructs 0^m is $O(F(\log n + \log c_{\max}))$. This can be proved using the technique of the expected multiplicative cost decrease with respect to $\min_{x \in P} cost(x)$. The proof is analogue to the single-commodity case $k = 1$ (see proof of Theorem 3 in [17]).

Now we bound the time until a minimum cut has been constructed. Once again we apply the method of the expected multiplicative cost decrease, now with respect to the flow value. Let x be the solution with the smallest flow value in $P \cap L$. Note that $\min_{x \in P \cap L} flow(x)$ does not increase during a run of GSEMO.

Consider a mutation step that selects x and performs an arbitrary 1-bit flip. Such a step is called a *good* step. The probability of a good step is lower bounded by $\Omega(1/F)$. By Proposition 1, $E(x)$ is a subset of an F/C^*-approximation of a minimum multicut, which can be obtained by including the remaining edges one by one. Therefore, a randomly chosen 1-bit flip decreases the minimum flow value in $P \cap L$ on average by a factor of at least $1 - 1/m$.

Hence, after N good steps, the expected minimum flow value is bounded from above by $(1 - 1/m)^N \cdot flow(x)$. Since $flow(x) \leq F \leq k \cdot C$, we obtain the upper bound $(1 - 1/m)^N \cdot k \cdot C$. Using the method of the multiplicative cost decrease the expected time until $x' \in L$ with $flow(x') = 0$ has been discovered is $O(Fm(\log n + \log c_{\max} + \log k))$. By Corollary 1, x' is an F/C^*-approximation of a minimum multicut. □

4 Analysis of DEMO

The upper bound given in Theorem 1 is polynomial as long as the weights are polynomially bounded with respect to the input size. For larger, i. e., exponential, weights the population size may become too large to obtain an F/C^*-approximation in expected polynomial time. To deal with this issue, we consider DEMO with an appropriate choice of ε such that the population size is always polynomially bounded with respect to the size of the given input.

To obtain the upper bound on the runtime of DEMO, we first consider the time until the search point 0^m has been included into the population and analyze the time to achieve an F/C^*-approximation afterwards.

Proposition 4. *Let $\varepsilon \leq 1/m$ and $x \in \{0,1\}^m$ a search point such that $cost(x) > 0$. Then there exists a 1-bit flip leading to a search point $x' \in \{0,1\}^m$ with $b_1(x') < b_1(x)$.*

Proof. Consider all 1-bit flips that remove a single edge from $E(x)$. Among all resulting search points, consider a point x' that minimizes $y' := cost(x')$. Let $y := cost(x)$.

The repeated removal of edges in $E(x)$ yields the search point 0^m. Let $\ell := |E(x)| \leq m$. Since y' was minimal, $y' \leq (1 - \frac{1}{\ell})y$ holds. Since $\varepsilon \leq \frac{1}{m} \leq \frac{1}{\ell}$ and $\ell \leq y$, we have

$$\begin{aligned}(1+\varepsilon)(1+y') &\leq 1 + \varepsilon + (1+\varepsilon)\left(1 - \frac{1}{\ell}\right) y \\ &\leq 1 + \frac{y}{\ell^2} + \left(1 + \frac{1}{\ell}\right)\left(1 - \frac{1}{\ell}\right) y = 1 + y .\end{aligned}$$

This implies

$$1 + \frac{\log(1+y')}{\log(1+\varepsilon)} \leq \frac{\log(1+y)}{\log(1+\varepsilon)} ,$$

and finally $b_1(x') < b_1(x)$. □

In the following, we bound the expected time until DEMO has produced the search point 0^m. Later on, we will show how the algorithm can proceed to obtain an F/C^*-approximation.

Lemma 2. *The expected time until DEMO working on the fitness function f includes the search point 0^m into the population is $O(m\varepsilon^{-2}(\log^2 n + \log^2 c_{\max}))$.*

Proof. The archiving strategy of DEMO guarantees that whenever a non-empty box becomes empty, another search point whose box dominates the considered box is included into the population. Therefore, $\min_{x \in P} b_1(x)$ will never increase during the run of the algorithm.

Since the population size is bounded by B, the probability of picking a search point $x \in P$ with minimal b_1-value is $\Omega(1/B)$. By Proposition 4, there exists at least one 1-bit flip leading to a search point x' with $b_1(x') < b_1(x)$. The probability to generate such a search point x' is $\Omega(1/m)$. After at most B such steps, the b_1-value is zero implying that we have found the search point 0^m. Hence, the expected time to include 0^m into the population is

$$O(B^2 m) = O(m\varepsilon^{-2} \log^2 C) = O(m\varepsilon^{-2}(\log^2 n + \log^2 c_{\max})).$$

This concludes the proof. □

To come up with an upper bound for DEMO, it is necessary to examine how the algorithm may progress from a solution $x \in L$ to a solution of $x' \in L$ with $b_2(x') < b_2(x)$. The following proposition points out that this is possible by carrying out a special 1-bit flip.

Proposition 5. *Let $\varepsilon \le 1/m$ and $x \in L$ a search point such that $flow(x) > 0$. Then there exists a 1-bit flip leading to a search point $x' \in L$ with $b_2(x') < b_2(x)$.*

Proof. By Corollary 2, there exists at least one 1-bit flip leading to a search point $x' \in L$ with $flow(x') < flow(x)$. Among all such search points, consider a point x' that minimizes $y' := flow(x')$. Let $y := flow(x)$.

The repeated application of Corollary 2 yields an F/C^*-approximation $E(x^*)$ of a minimum multicut of G. Let $\ell := |E(x^*)| - |E(x)| \le m$. Since y' was minimal, $y' \le (1 - \frac{1}{\ell})y$ holds. Since $\varepsilon \le \frac{1}{m} \le \frac{1}{\ell}$ and $\ell \le y$, we have $b_2(x') < b_2(x)$ by the same calculation as in the proof of Proposition 4. □

Finally, we are able to prove the following theorem which shows that the expected runtime of DEMO with an appropriate choice of ε is always polynomially bounded with respect to the given input.

Theorem 2. *Choosing $\varepsilon \le 1/m$, the expected time until DEMO working on the fitness function f constructs an F/C^*-approximation of a minimum multicut is $O(m\varepsilon^{-2}(\log^2 n + \log^2 c_{\max}))$.*

Proof. Due to Lemma 2 the search point $0^m \in L$ has been included into the population after an expected number of $O(m\varepsilon^{-2}(\log^2 n + \log^2 c_{\max}))$ steps. Hence, it is sufficient to consider the search process after having found a search point $x \in L$.

The archiving strategy of DEMO guarantees that whenever a non-empty box becomes empty, another search point whose box dominates the considered box is included into the population. Moreover, the tie-break rule ensures that a non-empty box with a search point $x \in P \cap L$ will never exchange that search point for a search point $x' \notin L$. Therefore, $\min_{x \in P \cap L} b_2(x)$ will never increase during the run of the algorithm.

Since the population size is bounded by B, the probability of picking a search point $x \in L$ with minimal b_2-value among the search points in L is $\Omega(1/B)$. By Proposition 5, there exists at least one 1-bit flip leading to a search point

$x' \in L$ with $b_2(x') < b_2(x)$. The probability to generate such a search point x' is $\Omega(1/m)$. After at most B such steps, the b_2-value is zero implying that we have found a multicut. Since $x' \in L$, this multicut is an F/C^*-approximation of a minimum cut. Hence, the expected time to obtain an F/C^*-approximation of a minimum multicut is

$$O(B^2 m) = O(m\varepsilon^{-2} \log^2 C) = O(m\varepsilon^{-2}(\log^2 n + \log^2 c_{\max}))\,.$$

This concludes the proof. □

5 Conclusions

The multicut problem is an NP-hard generalization of the minimum cut problem. We have shown how the correlation between flows and cuts can be used to come up with efficient evolutionary algorithms for the approximation of the minimum multicut problem. Our multi-objective approach using flow computations and the concept of ε-dominance is able to achieve a k-approximation in expected polynomial time. Further studies will consider how the theoretical results obtained in this paper can be used to come up with good evolutionary algorithms for the multicut problem by using our model in well-known evolutionary multi-objective algorithms.

Acknowledgements

The authors thank Martin Skutella for helpful discussions on the topic of this paper.

References

1. Ahuja, R.K., Magnati, T.L., Orlin, J.B.: Network flows: theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs (1993)
2. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM 45(3), 501–555 (1998)
3. Bentz, C., Costa, M.-C., Létocard, L., Roupin, F.: Erratum to M.-C. Costa, L. Létocard and F. Roupin: Minimal multicut and maximal integer maxiflow: A survey. European Journal of Operational Research 162(1), 55–69 (2005); European Journal of Operational Research 177(2), 1312 (2007)
4. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do additional objectives make a problem harder? In: Proc. of the 9th Genetic and Evolutionary Comp. Conference (GECCO 2007), pp. 765–772. ACM Press, New York (2007)
5. Cheriyan, J., Karloff, H., Rabani, Y.: Approximating directed multicuts. In: Proc. of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS 2001), pp. 320–328 (2001)
6. Chuzhoy, J., Khanna, S.: Polynomial flow-cut gaps and hardness of directed cut problems. In: Proc. of the 39th Annual ACM Symposium on Theory of Computing (STOC 2007), pp. 179–188 (2007)

7. Costa, M.-C., Létocard, L., Roupin, F.: Minimal multicut and maximal integer maxiflow: A survey. European Journal of Operational Research 162(1), 55–69 (2005)
8. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. SIAM Journal of Computing 23, 864–894 (1994)
9. Doerr, B., Happ, E., Klein, C.: Crossover is provably useful in evolutionary computation. In: Proc. of the 10th Genetic and Evolutionary Computation Conference (GECCO 2008). ACM Press, New York (to appear, 2008)
10. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. In: Proc. of the 9th Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 797–804 (2007)
11. Garg, N., Vazirani, V.V., Yannakakis, M.: Approximate max-flow min-(multi)cut theorems and their applications. In: Proc. of the 25th Annual ACM Symposium on Theory of Computing (STOC 1993), pp. 698–707 (1993)
12. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC 2003), pp. 1918–1925 (2003)
13. Horoba, C., Neumann, F.: Benefits and drawbacks for the use of ε-dominance in evolutionary multi-objective optimization. In: Proc. of the 10th Genetic and Evolutionary Computation Conference (GECCO 2008). ACM Press, New York (to appear, 2008)
14. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation 10(3), 263–282 (2002)
15. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. IEEE Transactions on Evolutionary Computation 8(2), 170–182 (2004)
16. Neumann, F.: Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. European Journal of Operational Research 181(3), 1620–1629 (2007)
17. Neumann, F., Reichel, J., Skutella, M.: Computing minimum cuts by randomized search heuristics. In: Proc. of the 10th Genetic and Evolutionary Computation Conference (GECCO 2008) (2008); (to appear, available as Technical Report CI-242/08, Collaborative Research Center 531, Technical University of Dortmund)
18. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Natural Computing 5(3), 305–319 (2006)
19. Neumann, F., Wegener, I.: Randomized local search, evolutionary algorithms and the minimum spanning tree problem. Theor. Comp. Sci. 378(1), 32–40 (2007)
20. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. Journal of Mathematical Modelling and Algorithms, 349–366 (2004)
21. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003)
22. Witt, C.: Worst-case and average-case approximations by simple randomized search heuristics. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 44–56. Springer, Heidelberg (2005)

Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation

Pietro S. Oliveto[1,⋆] and Carsten Witt[2,⋆⋆]

[1] Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Birmingham, UK
[2] Fakultät für Informatik, LS 2, Technische Universität Dortmund, Dortmund, Germany

Abstract. Drift analysis is a powerful tool used to bound the optimization time of evolutionary algorithms (EAs). Various previous works apply a drift theorem going back to Hajek in order to show exponential lower bounds on the optimization time of EAs. However, this drift theorem is tedious to read and to apply since it requires two bounds on the moment-generating (exponential) function of the drift. A recent work identifies a specialization of this drift theorem that is much easier to apply. Nevertheless, it is not as simple and not as general as possible. The present paper picks up Hajek's line of thought to prove a drift theorem that is very easy to use in evolutionary computation. Only two conditions have to be verified, one of which holds for virtually all EAs with standard mutation. The other condition is a bound on what is really relevant, the drift. Applications show how previous analyses involving the complicated theorem can be redone in a much simpler and clearer way. Therefore, the simplified theorem is also a didactical contribution to the runtime analysis of EAs.

1 Introduction

Theoretical studies of the computational complexity of Evolutionary Algorithms (EAs) have appeared since the 1990s (see Oliveto, He and Yao [1]). Since then various mathematical techniques for the analysis of EAs have been constructed. An overview of many important tools can be found in Wegener [2].

Recently *drift analysis*, a technique that goes back to the 1940s (cf. the introduction in [3]), was introduced for the analysis of EAs by He and Yao [4,5]. The authors concentrated on the obtainment of both lower and upper bounds on the expected runtime of EAs. Concerning lower bounds, Giel and Wegener [6] point out that a drift theorem on the success probability may also be obtained rather than only the expected waiting time. In this form the drift theorem has been used several times (e. g., Giel and Wegener [6] for maximum matching, Oliveto, He and Yao [7] for vertex cover, Friedrich, Oliveto, Sudholt and Witt [8] for

⋆ The first author was supported by an EPSRC grant (EP/C520696/1).
⋆⋆ The second author was supported by the Deutsche Forschungsgemeinschaft (DFG) as a part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

analyzing population-based EAs with diversity mechanisms etc.) to prove exponential lower bounds on optimization times that even hold with probabilities exponentially close to 1. Although the mentioned drift theorem has turned out to be very useful, it often leads to tedious and complicated calculations. This seems to be the price to pay for the sake of keeping the drift theorem as general as possible. However, by considering the characteristics of the stochastic processes defined by EAs, it is possible to derive conditions which are more restrictive but considerably easier to verify. In fact, with similar motivations, Happ, Johannsen, Klein and Neumann [9] have recently introduced a simplified drift theorem.

In this paper we present a further simplification of the drift theorem which is particularly suited for the analysis of EAs. Our proof resembles the argumentation used by Hajek to verify the conditions of its complicated but general theorem. It seems that, to a certain extent, many applications of the complicated theorem rely on a historical accident. Hajek himself states simpler but more restrictive conditions which he claims to be useful in applications. We only slightly tweak these conditions to make them even easier to verify in the analysis of EAs.

The rest of the paper is structured as follows. Section 2 presents some background on drift analysis and the simplified drift theorem. Afterwards, we study some exemplary applications to show the strength and elegance of the new approach. Section 3 contains a warm-up example. In Section 4 we show that the simplified drift theorem can also be used in the setting of Happ et al. [9] and that even significantly stronger results are obtained with shorter proofs. In Section 5 we study the maximum matching problem as an advanced application to show that proofs are considerably simplified. We finish with some conclusions.

2 Previous Work and the Simplified Drift Theorem

Hajek introduced the following theorem to provide a flexible technique for proving the stability of processes frequently encountered in queuing systems [3]. Since then, it has been restated in different forms several times (e. g., He and Yao [4] and Giel and Wegener [6]) to adapt it for the analysis of EAs. With the aim of proving exponential lower bounds on first hitting times, usually four conditions to be fulfilled are listed. Interestingly, in essence, there is only a single inequality, namely a bound on the moment-generating function of the one-step drift, to be checked for the final statement of the theorem to hold. By analyzing the original proof, it follows that the remaining conditions can be either rephrased or removed. In particular, there is no need for the following values $\lambda(\ell)$ and $D(\ell)$ to be constant or $p(\ell)$ to be polynomial. In any case, for the theorem to be meaningful, it has to be assured that $D(\ell)$ is defined.

Theorem 1 (Hajek [3]). *Let* $X_0, X_1, X_2, \ldots$ *be the random variables describing a Markov process over a state space* S *and* $g\colon S \to \mathbb{R}_0^+$ *a function mapping each state to a non-negative real number. Pick two real numbers* $a(\ell)$ *and* $b(\ell)$ *depending on a parameter* ℓ *such that* $0 \le a(\ell) < b(\ell)$ *holds. Let* $T(\ell)$ *be the* *random variable denoting the earliest point in time $t \geq 0$ such that $g(X_t) \leq a(\ell)$ holds. If there are $\lambda(\ell) > 0$ and $p(\ell) > 0$ such that the condition*

$$E\bigl(e^{-\lambda(\ell)\cdot(g(X_{t+1})-g(X_t))} \mid a(\ell) < g(X_t) < b(\ell)\bigr) \;\leq\; 1 - \frac{1}{p(\ell)} \textit{ for all } t \geq 0 \quad (*)$$

holds then for all time bounds $L(\ell) \geq 0$

$$\mathrm{Prob}\bigl(T(\ell) \leq L(\ell) \mid g(X_0) \geq b(\ell)\bigr) \;\leq\; e^{-\lambda(\ell)\cdot(b(\ell)-a(\ell))} \cdot L(\ell) \cdot D(\ell) \cdot p(\ell),$$

where $D(\ell) = \max\bigl\{1, E\bigl(e^{-\lambda(\ell)\cdot(g(X_{t+1})-b(\ell))} \mid g(X_t) \geq b(\ell)\bigr)\bigr\}$.

In the typical applications of Theorem 1 cited above, the main drift Condition $(*)$ is proved with $p(\ell)$ being a polynomial. Having accomplished this, it often easily follows that $D(\ell)$ does not grow with ℓ. The values $a(\ell)$ and $b(\ell)$ are frequently chosen linear in the dimension of the search space n such that $b(\ell) - a(\ell) = \Omega(n)$ and $\ell = \Omega(n)$ while $\lambda(\ell)$ is chosen constant. Consequently, choosing $L(\ell) = 2^{cn}$, where c is a sufficiently small constant, the final statement of the theorem boils down to $\mathrm{Prob}(T(\ell) \leq 2^{cn}) \leq 2^{-\Omega(n)}$. This is as desired: even given exponential time, the probability of finding the optimum (i. e., $g(X_t) \leq a$) is exponentially small w. r. t. the problem dimensionality.

Happ et al. [9] present a simplified version of the drift theorem called "Global Gambler's Ruin" with conditions that are much easier to check. The main simplification introduced to prove Condition $(*)$ of the original theorem is as follows: assuming $S = \mathbb{N}_0$ and $g = \mathrm{id}$, they demand the existence of a constant $\delta > 1$ such that, given $X_t = i$, the condition $\mathrm{Prob}(X_{t+1} = i + j) \geq \delta^j \,\mathrm{Prob}(X_{t+1} = i - j)$ holds for all $j \geq 1$. Intuitively, this means that for every step length j, there is a bias (drift) towards increasing the state by j compared to decreasing it by j; moreover, this bias increases exponentially w. r. t. j. In an application to an EA with fitness-proportional selection, it turns out that the new condition is relatively easy to verify. The drawback is that $a(\ell)$ and $b(\ell)$ have to be chosen carefully to establish the exponential bias δ^j for all j. Moreover, the new theorem by Happ et al. [9] contains an additional condition on – in essence – the moment-generating function $E(\delta^{-(X_{t+1}-X_t)} \mid X_t \geq b(\ell))$ in order to bound the value $D(\ell)$ of the original theorem. Despite being relatively easy to verify, both conditions seem stronger than needed for our purpose.

Our main contribution is another simplification of the drift theorem, which is particularly suited for the stochastic processes described by evolutionary algorithms and even easier to apply than the version by Happ et al. [9]. With the aim of proving that the process does not pass the interval $[a, b]$ in exponential time if started above state b, we intuitively need the following two conditions:

– Assuming to be in the interval at time t, there must be a drift, an expected displacement, towards increasing the state, more precisely, there must be some constant $\varepsilon > 0$ such that $\sum_{j\in\mathbb{Z}} j \cdot \mathrm{Prob}(X_{t+1} = i + j \mid X_t = i) \geq \varepsilon$ for all i in the interval. There seems to be no need for the drift to be bounded in the same manner for every j or even to increase with j.

– Drift alone is not enough. Considering exponentially long phases, the probability must be exponentially small to leave the interval towards the optimum using large jumps. The random step length towards the optimum has to exhibit an exponential decay. This follows from $\mathrm{Prob}(X_{t+1} = i - j \mid X_t = i) \leq 1/(1+\delta)^{j-r}$ for constants $\delta, r > 0$ and all $i > a$, i. e., within and outside the interval. We will see that this second condition always holds for standard bit flip mutations.

Besides, we will need a technical condition regarding the absolute convergence of the power series appearing in the following proof. Since we usually consider finite search spaces, we restrict the state space of the Markov process to $\{0, 1, \dots, N\}$ for an arbitrarily large integer N and obtain such convergence for free. Weaker conditions could be proven if applications in infinite search spaces are desired.

We are ready to state and prove our simplified drift theorem. Note an additional difference to the version by Happ et al. [9]: a and b do not need to be linear in the dimension of the search space.

Theorem 2 (Simplified Drift Theorem). *Let X_t, $t \geq 0$, be the random variables describing a Markov process over the state space $S := \{0, 1, \dots, N\}$ and denote $\varDelta_t(i) := (X_{t+1} - X_t \mid X_t = i)$ for $i \in S$ and $t \geq 0$. Suppose there exist an interval $[a, b]$ of the state space and three constants $\delta, \varepsilon, r > 0$ such that for all $t \geq 0$*

1. *$E(\varDelta_t(i)) \;\geq\; \varepsilon$ for $a < i < b$*
2. *$\mathrm{Prob}(\varDelta_t(i) = -j) \;\leq\; 1/(1+\delta)^{j-r}$ for $i > a$ and $j \geq 1$*

then there is a constant $c^* > 0$ such that for $T^* := \min\{t \geq 0\colon X_t \leq a \mid X_0 \geq b\}$ it holds $\mathrm{Prob}(T^* \leq 2^{c^*(b-a)}) = 2^{-\varOmega(b-a)}$.

Proof. Define $\ell := b - a$ and note that $\ell \leq N$. We will apply Theorem 1 for suitable choices of its variables. Some of these might depend on the parameters N and ℓ. As will be shown later, only $L(\ell)$ depends on a parameter (namely ℓ), hence we will omit any indices from the remaining parameters λ, D and p. Moreover, we set $g := \mathrm{id}$. The following argumentation is also inspired by Hajek's work [3].

Fix $t \geq 0$ and some i such that $a < i < b$ and denote $p_j := \mathrm{Prob}(\varDelta_t(i) = j)$. To prove Condition $(*)$, it is sufficient to identify a constant $\lambda > 0$ such that

$$S(\lambda) := \sum_{j \in \mathbb{Z}} e^{\lambda j} p_{-j} < 1.$$

Using the series expansion for $e^{\lambda j} = \sum_{k=0}^{\infty} (\lambda j)^k / k!$, we have

$$S(\lambda) \;=\; \sum_{j \in \mathbb{Z}} \sum_{k=0}^{\infty} \frac{(\lambda j)^k}{k!} p_{-j} \;=\; 1 + \sum_{j \in \mathbb{Z}} (\lambda j) \cdot p_{-j} + \sum_{k=2}^{\infty} \sum_{j \in \mathbb{Z}} \frac{(\lambda j)^k}{k!} p_{-j},$$

where all series converge absolutely for any $\lambda > 0$ since $p_j = 0$ for $|j| > N$; however, their limits might depend on N. Identifying $E(\varDelta_t(i))$ and using the first condition of the theorem, the bound on the drift, we obtain for all $\gamma \geq \lambda$

$$S(\lambda) \;\leq\; 1 - \lambda E(\varDelta_t(i)) + \frac{\lambda^2}{\gamma^2} \sum_{k=2}^{\infty} \sum_{j \in \mathbb{Z}} \frac{(\gamma j)^k}{k!} p_{-j} \;\leq\; 1 - \lambda\varepsilon + \lambda^2 \cdot \underbrace{\frac{\sum_{j \in \mathbb{Z}} e^{\gamma j} p_{-j}}{\gamma^2}}_{=:C(\gamma)}.$$

Given any $\gamma > 0$, choosing $\lambda := \min\{\gamma, \varepsilon/(2C(\gamma))\}$ results in

$$S(\lambda) \leq 1 - \lambda\varepsilon + \lambda \cdot \frac{\varepsilon}{2C(\gamma)} \cdot C(\gamma) = 1 - \frac{\lambda\varepsilon}{2} < 1$$

as desired. Choosing $\gamma := \ln(1 + \delta/2)$, which does not depend on ℓ and N, and exploiting the second condition yields

$$C'(\gamma) := \sum_{j\in\mathbb{Z}} e^{\gamma j} p_{-j} \leq \sum_{j\leq 0} (1 + \delta/2)^j + \sum_{j\geq 1} \frac{(1 + \delta/2)^j}{(1 + \delta)^{j-r}} \leq (1 + \delta)^r \left(2 + \frac{4}{\delta}\right),$$

hence $C(\gamma) \leq (1+\delta)^r(3+4/\delta)/\ln^2(1+\delta/2)$, which does not depend on ℓ and N either. Since, moreover, ε, δ, r do not depend on these parameters, neither will $C(\gamma)$, γ, λ, nor our bound on $S(\lambda)$. This establishes Condition $(*)$ of Theorem 1 for $p = O(1)$ and $\lambda = \Omega(1)$.

To bound the probability of a success within $L(\ell)$ steps, we still need a bound on $D = \max\{1, E(e^{-\lambda(X_{t+1}-b)} \mid X_t \geq b)\}$. Since $\lambda \leq \gamma$ and $X_t \geq b$, we have $D \leq E(e^{-\lambda(X_{t+1}-b)}) \leq E(e^{-\gamma(X_{t+1}-X_t)}) = \sum_{j\in\mathbb{Z}} e^{\gamma j} \cdot \text{Prob}(\Delta_t(i) = -j)$ for $i \geq b$. Note that the calculation leading above to $C'(\gamma) = O(1)$ holds for arbitrary $i \geq a$ due to the second condition. Hence, $D = O(1)$, which does not depend on a parameter either. Altogether, we have $e^{-\lambda\ell}Dp = 2^{-\Omega(\ell)} = 2^{-\Omega(b-a)}$. Choosing $L(\ell) = 2^{c^*(b-a)}$ for some sufficiently small constant $c^* > 0$, Theorem 1 yields $\text{Prob}(T(\ell) \leq L(\ell)) \leq L(\ell) \cdot 2^{-\Omega(b-a)} = 2^{-\Omega(b-a)}$, which proves the theorem. □

Our drift theorem can easily be applied to Randomized Local Search (RLS) on the search space $\{0, 1\}^n$, which flips only one bit per iteration. Then Condition 2 is trivial and the theorem resembles the well-known *Gambler's Ruin* Theorem (see also [9]). However, the generalized drift technique was previously used to obtain lower bounds on the first hitting time of the (1+1)-EA, which can flip several bits in a step. Then the original Gambler's Ruin Theorem does not apply. For maximization problems, the (1+1)-EA is defined as follows.

(1+1)-EA

```
– Choose uniformly at random an initial bit string $x \in \{0, 1\}^n$;
– Repeat the following steps until a termination criterion is satisfied:
    1. Create $x'$ by flipping each bit in $x$ with probability $p_m := 1/n$;
    2. Replace $x$ with $x'$ if $f(x') \geq f(x)$;
```

In the rest of the paper we will show that proofs regarding lower bounds on the runtime of the (1+1)-EA that hold with overwhelming probability $1 - 2^{-\Omega(b-a)}$ are really easy to obtain by using the proposed drift theorem. Our proofs are universal enough to apply, after some tiny changes, also for RLS. This is however not everywhere made explicit due to space limitations.

3 An Application for the (1+1)-EA

In this section we present a first application of Theorem 2. We choose the Needle-in-a-haystack function which is well known to be hard for EAs [10] and show that the (1+1)-EA is even at distance almost $n/2$ from its optimum for an exponential number of steps. The whole search space consists of a plateau except for one point representing the global optimum. W. l. o. g. we choose the optimum to be the point represented by the bit string of all ones. The function is the following:

$$\textsc{Needle}_n(x) = \begin{cases} 1 \text{ if } x = 1^n, \text{ and} \\ 0 \text{ otherwise.} \end{cases}$$

Theorem 3. *Let $\eta > 0$ be constant. Then there is a constant $c > 0$ such that with probability $1 - 2^{-\Omega(n)}$ the (1+1)-EA on* $\textsc{Needle}_n$ *creates only search points with at most $n/2 + \eta n$ ones in 2^{cn} steps.*

Proof. Let X_t denote the number of zeroes in the bit string at time step t. We set $a := n/2 - 2\gamma n$ and $b := n/2 - \gamma n$, where $\gamma := \eta/2$. Such a value for b is suitable because by Chernoff bounds the probability that the initial bit string has less than $n/2 - \gamma n$ zeroes is $2^{-\Omega(n)}$. Now we use the proposed simplified drift theorem for the rest of the proof. It therefore remains to check that the two conditions of Theorem 2 hold.

Given a string in state $i < n/2 - \gamma n$, i. e., with i zeroes, let $\Delta(i)$ denote the random increase of the number of zeroes. Condition 1 holds if $E(\Delta(i)) \geq \varepsilon$ for some constant $\varepsilon > 0$. Since the (1+1)-EA flips 0-bits and 1-bits independently, an expected number of i/n 0-bits and $(n-i)/n$ 1-bits is flipped. Hence,

$$E(\Delta(i)) \;=\; \frac{n-i}{n} - \frac{i}{n} \;=\; \frac{n-2i}{n} \;\geq\; 2\gamma$$

So we can choose $\varepsilon = 2\gamma$.

Condition 2 is: $\mathrm{Prob}(\Delta(i) = -j) \leq 1/(1+\delta)^{j-r}$. In order to reach state $i - j$ from state i, at least j bits have to flip. Hence $\mathrm{Prob}(\Delta(i) = -j) \leq \binom{n}{j}(1/n)^j \leq 1/j! \leq (1/2)^{j-1}$, which proves the condition for $\delta = 1$ and $r = 1$ even independently of i and of selection. So from Theorem 2 it follows for a constant $c^* > 0$ that the global optimum is found in $2^{c^*(b-a)} = 2^{cn}$ steps, where $c := c^*(b-a)/n > 0$ is a different constant, with probability at most $2^{-\Omega(b)} = 2^{-\Omega(n)}$. □

4 An Application for the (1+1)-EA with Fitness-Proportional Selection

Recently Happ et al. [9] have presented a simplified drift theorem called *Global Gambler's Ruin.* They introduced the new theorem to prove that the (1+1)-EA using fitness-proportional selection requires exponential runtime for optimizing Onemax and linear functions in general. The algorithm works as follows:

(1+1)-EA with Fitness-proportional Selection ((1+1)-EA$_{\mathbf{prop}}$)

```
– Choose uniformly at random an initial bit string $x \in \{0,1\}^n$;
– Repeat the following steps until a termination criterion is satisfied:
    1. Create $x'$ by flipping each bit in $x$ with probability $p_m := 1/n$;
    2. Replace $x$ with $x'$ with probability $f(x')/(f(x') + f(x))$;
```

A function $f\colon \{0,1\}^n \to \mathbb{R}$ is *linear* if it can be written as $f(x_1, \ldots, x_n) = w_0 + w_1x_1 + \cdots + w_nx_n$ with coefficients $w_i \geq 0$, $0 \leq i \leq n$. In the special case

$w_1 = \cdots = w_n = 1$ and $w_0 = 0$ we obtain the OneMax function counting the number of ones of the bit string. Concerning linear functions, Happ et al. [9] prove that with overwhelming probability only search points with at most $0.97n$ ones are created by the (1+1)-EA$_{\text{prop}}$ after an exponential number of steps. We show that Theorem 2 can be used for this purpose and that it can lead to significantly stronger results. We remark that the following proof also holds for fitness-proportional RLS, where the stronger statement is already known [9].

Theorem 4. *Let $0 < \eta \leq 1/4$ and η be constant. Then there is a constant $c > 0$ such that with probability $1 - 2^{-\Omega(n)}$ the (1+1)-EA$_{\text{prop}}$ for linear functions (for* OneMax*) only creates search points with at most $2n/3 + \eta n$ (resp. at most $n/2 + \eta n$) ones in 2^{cn} steps.*

Proof. Setting $a := n/3 - 2\gamma n$ and $b := n/3 - \gamma n$, where $\gamma := \eta/2 \leq 1/8$, and given a current number of $a < i < b$ zeroes, let $\Delta(i)$ and $\Delta^{\text{sel}}(i)$ denote the random change in this number before and after selection, respectively. Using the arguments from the proof of Theorem 3, we get $E(\Delta(i)) = (n-2i)/n \geq 1/3 + 2\gamma$.

$E(\Delta(i))$ is mostly determined by small steps. Choosing $r := \gamma n/4$, define $\mathbb{1}_r := \mathbb{1}\{|\Delta(i)| \leq r\}$ as the indicator r. v. for the event $|\Delta(i)| \leq r$. Since flipping at least k bits in a step has probability at most $1/k!$ and at most n bits flip,

$$E\big(\Delta(i) \cdot \mathbb{1}_r\big) \;\geq\; E(\Delta(i)) - \frac{1}{(\gamma n/4)!} \cdot n \;=\; E(\Delta(i)) - 2^{-\Omega(n)}$$

and accordingly for $E(\Delta^{\text{sel}}(i))$. By concentrating on steps of length at most r, we therefore introduce only an exponentially small error.

$\Delta(i)$ can be decomposed according to $\Delta(i) := \Delta^+(i) - \Delta^-(i)$, where $\Delta^+(i) := \Delta(i) \cdot \mathbb{1}\{\Delta(i) > 0\}$ and $\Delta^-(i) := -\Delta(i) \cdot \mathbb{1}\{\Delta(i) < 0\}$. By considering only the flipping 0-bits, we get $E(\Delta^-(i)) \leq 1/3 - \gamma$. Using $E(\Delta(i)) \geq 1/3 + 2\gamma$, we obtain $E(\Delta^+(i))/E(\Delta^-(i)) = (E(\Delta(i)) + E(\Delta^-(i)))/E(\Delta^-(i)) \geq 2 + 3\gamma$.

We get a lower bound on $E(\Delta^{\text{sel}}(i))$ by weighting $\Delta^-(i)$ with upper bounds (here 1) on the selection probability and $\Delta^+(i)$ with lower bounds. For the lower bounds, we pessimistically assume all zeroes of the current string x to have coefficients 0 and $w_0 = 0$. Then $f(x') \leq f(x)$ for all offspring x' of x and the selection probability is at least $\frac{f(x')}{f(x')+f(x)} \geq \frac{f(x')}{2f(x)}$, which is linear w. r. t. $f(x')$. We assume at most r flipping bits. If a random subset of r out of $n - i \geq \frac{n}{2}$ ones flips, each bit flips with probability $\frac{r}{n-i}$. Using the linearity of expectation and of f, the expected offspring value $e(x')$ is at least $f(x)(1 - \frac{r}{n-i})$. Thus, using the law of total probability, the selection probability for the random x' is at least $e(x')/(2f(x)) \geq \frac{1}{2} - \frac{r}{2(n-i)} \geq \frac{1}{2} - \frac{\gamma}{4}$. Since the bound is independent of $\Delta(i) \cdot \mathbb{1}_r$,

$$\begin{aligned}
E(\Delta^{\text{sel}}(i)) \;&\geq\; \left(\frac{1}{2} - \frac{\gamma}{4}\right) E(\Delta^+(i) \cdot \mathbb{1}_r) - E(\Delta^-(i) \cdot \mathbb{1}_r) - 2^{-\Omega(n)} \\
&\geq\; \left(\frac{1}{2} - \frac{\gamma}{4}\right) (2 + 3\gamma) E(\Delta^-(i) \cdot \mathbb{1}_r) - E(\Delta^-(i) \cdot \mathbb{1}_r) - 2^{-\Omega(n)} \\
&\geq\; \left(\gamma - \frac{3\gamma^2}{4}\right) E(\Delta^-(i) \cdot \mathbb{1}_r) - 2^{-\Omega(n)} \;\geq\; \frac{29\gamma}{32} \cdot \frac{1}{36} - 2^{-\Omega(n)} \;\geq\; \frac{\gamma}{40}
\end{aligned}$$

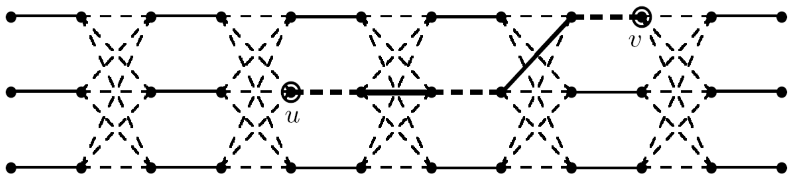


Fig. 1. The $G_{h,\ell}$ graph (in this case $h = 3$ and $\ell = 11$) with an almost perfect matching and its augmenting path between u and v

for n large enough, where we have used that $\gamma \leq 1/8$ along with $E(\Delta^-(i)) \geq (1/3 - 2\gamma)(1 - 1/n)^{n-1} \geq (1/3 - 2\gamma)/e \geq 1/36$, which follows by considering only 1-bit mutations. This bounds the drift for general linear functions by a constant.

With OneMax, the situation is even simpler. Since then f equals the number of ones, we can bound the probability of accepting a string x' with up to r more ones than x by $f(x')/(2f(x)) \leq (f(x) + r)/(2f(x)) \leq f(x)(1 + \gamma/2)/(2f(x)) = 1/2 + \gamma/4$ using $f(x) \geq n/2$. Setting $a := n/2 - 2\gamma n$ and $b := n/2 - \gamma n$, a similar calculation as in the third paragraph of this proof yields $E(\Delta^+(i))/E(\Delta^-(i)) \geq 1 + 2\gamma$. Finally, we obtain $E(\Delta^{\text{sel}}(i)) \geq ((1/2 - \gamma/4)(1 + 2\gamma) - (1/2 + \gamma/4)) \cdot E(\Delta^-(i)) - 2^{-\Omega(n)} \geq \gamma/100$ for n large enough in the same manner as above.

The rest of the argumentation, in particular the proof of Condition 2 of Theorem 2 carries over from the proof of Theorem 3. □

5 An Advanced Application: Maximum Matching

Giel and Wegener [6] considered the graph depicted in Figure 1 to prove that the (1+1)-EA has an expected runtime which is exponential in the number of graph edges for the well known *maximum matching* problem in the worst case. One of the crucial parts of their proof is represented by the following theorem.

Theorem 5. *Starting with an almost perfect matching with an augmenting path of length ℓ, the probability that the (1+1)-EA finds the perfect matching of the $G_{h,\ell}$ graph within $2^{c\ell}$ steps, $c > 0$ an appropriate constant, is bounded by $2^{-\Omega(\ell)}$ if $h \geq 3$.*

Proof. An *almost perfect matching* is just one fitness level away from the global optimum. In order to find the maximum matching, the edges of the only augmenting path in the graph have to be either inverted or the path has to be shortened to its minimum (i. e., three adjacent edges not belonging to the matching are obtained). If the latter case happens, then the extra edge may be added by just using one bit flip. Given an almost perfect matching, a move of length $j = 1$ occurs if at least two adjacent edges flip on either side of the augmenting path. The augmenting path may be *lengthened* or *shortened*. In the former case the process drifts away from the optimum while in the latter case it heads towards it. To apply Theorem 2, we set $a := 0$, the minimum augmenting path length and $b := \ell - 1$ where ℓ is its maximum length.

Usually there are $2h$ edges adjacent to the augmenting path, h at each side, that flipped together with the first edge belonging to the path would *lengthen* it. However, if the augmenting path starts at the beginning of the graph (or at the other end), then there are only h such edges (actually this shows that the length of the augmenting path is not enough to describe the underlying Markov process exactly, yet it gives good enough bounds). In this case, the probability of performing a move of length 1 *lengthening* the augmenting path of length i is only bounded by $p_1(i) \geq (h/m^2)(1-1/m)^{m-2}$, where m is the number of edges of the graph. On the other hand, the probability to shorten the augmenting path with a move of length 1 is bounded from above by $p_{-1}(i) \leq (2/m^2)(1-1/m)^{m-2}+3/m^4$ (see [6]). Since most other mutations of the (1+1)-EA due to worse fitness will be rejected in this setting, we use the condition R that a step is *relevant*, meaning it is accepted and changes the current state. The probability p_{rel} of a relevant step is bounded by $(1/m^2)(1-1/m)^{m-2} \leq p_{\text{rel}} \leq (2h+2)/m^2$.

Let $R(i) = (\Delta(i) \mid R)$ denote the random increase of the path length in relevant steps for a current length i. It suffices to concentrate on the contribution of steps of length 1, i. e., we consider $R_1(i) := R(i) \cdot \mathbb{1}\{|R(i)| \leq 1\}$. We obtain

$$E(R_1(i)) = \frac{p_1(i)}{p_{\text{rel}}} - \frac{p_{-1}(i)}{p_{\text{rel}}} \geq \frac{h-2-O(m^{-2})}{2h+2} \geq \frac{1}{8} - O(m^{-2})$$

since $h \geq 3$ while the unconditional decrease $\Delta^-_{>1}(i) = -\Delta(i) \cdot \mathbb{1}\{\Delta(i) < -1\}$, for negative steps of length greater than 1, in expectation is at most

$$E(\Delta^-_{>1}(i)) \leq \sum_{j=2}^{\infty} j \cdot p_{-j}(i) \leq \sum_{j=2}^{\infty} j \cdot (j+1)\frac{1}{m^{2j}} \leq \frac{6}{m^4} + \sum_{j=3}^{\infty} \frac{2m^2}{m^{2j}} = O(m^{-4})$$

because $p_{-j} \leq (j+1)/m^{2j}$ [6].

Hence, the total conditional drift is

$$E(R(i)) \geq E(R_1(i)) - \frac{E(\Delta^-_{>1}(i))}{p_{\text{rel}}} \geq \frac{1}{8} - O(m^{-2}) - O(m^{-4}) \cdot em^2 = \frac{1}{8} - O(m^{-2})$$

and Condition 1 is proved. Condition 2, with $\delta = 1$ and $r = 3$, follows from

$$\frac{p_{-j}}{p_{\text{rel}}} \leq \min\left\{1, \frac{j+1}{m^{2j}} \cdot em^2\right\} \leq \min\left\{1, \frac{1}{m^{2j-7}}\right\} \leq \left(\frac{1}{2}\right)^{j-3}$$

for $m \geq 2$. From Theorem 2, the proof follows. □

The bounds on $p_j(i)$ by Giel and Wegener [6] do not imply $p_j(i) \geq p_{-j}(i)$ for every j, hence the theorem by Happ et al. [9] does not apply with these bounds. Without further work on the bounds for $p_j(i)$, it is crucial but also sufficient to focus on the effect of steps of length 1.

6 Conclusion

A simplified drift-analysis theorem has been introduced for proving lower bounds on the runtime of EAs that hold with high probability. The two hypotheses of the theorem are easy to check for stochastic processes such as those described by EAs. The *first condition* holds if the distance to the optimum increases in expectation by at least a constant amount. In other terms, there is a *drift* leading away from the optimum. The *second condition* describes an *exponential decay* in the probabilities of advancing towards the optimum that depends on the step size. Such a condition is trivially fulfilled for the (1+1)-EA with standard mutation and many other EAs with a mutation operator that exhibits enough locality. The simplified drift theorem allowed us to redo previous analyses with significantly reduced effort.

For scenarios where bounding the drift directly is more intricate a corollary of the simplified theorem might be mentioned. It is sufficient to decompose the drift into the effects of steps of a given length and to prove a bias leading away from the optimum for every step length. In fact, also Happ et al. [9] exploited a similar idea. Our corollary, though, seems to be easier to verify since we do not require the bias to increase with the step length. Moreover, compared to the latter work, we do not require that the length of the drift interval $[a, b]$ is $\Omega(n)$. Our generalization is necessary, for example, in the study by Friedrich et al. [8] where $b - a = \sqrt[3]{n}$. To the best of our knowledge all previous applications of drift analysis to evolutionary computation can be proven in a considerably simpler shape with the proposed simplified drift theorem. As a result, not only is Theorem 2 considered as an important didactical contribution to the runtime analysis of EAs, but we also believe it will turn out to be useful in future work.

References

1. Oliveto, P.S., He, J., Yao, X.: Computational complexity analysis of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing 4, 281–293 (2007)
2. Wegener, I.: Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In: Sarker, R., Mohammadian, M., Yao, X. (eds.) Evolutionary Optimization. Kluwer Academic Publishers, Dordrecht (2001)
3. Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. Advances in Applied Probability 13, 502–525 (1982)
4. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence 127, 57–85 (2001)
5. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3, 21–35 (2004)
6. Giel, O., Wegener, I.: Evolutionary algorithms and the maximum matching problem. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 415–426. Springer, Heidelberg (2003)
7. Oliveto, P.S., He, J., Yao, X.: Evolutionary algorithms and the vertex cover problem. In: Proc. of CEC 2007, pp. 1430–1438 (2007)
8. Friedrich, T., Oliveto, P.S., Sudholt, D., Witt, C.: Theoretical analysis of diversity mechanisms for global exploration. In: Proc. of GECCO 2008 (to appear, 2008)
9. Happ, E., Johannsen, D., Klein, C., Neumann, F.: Rigorous analyses of fitness-proportional selection for optimizing linear functions. In: Proc. of GECCO 2008 (to appear, 2008)
10. Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. Evolutionary Computation 7(2), 173–203 (1999)

Ignoble Trails - Where Crossover Is Provably Harmful

J. Neal Richter[1], Alden Wright[2], and John Paxton[1]

[1] Montana State University,
richter@cs.montana.edu, paxton@cs.montana.edu
[2] University of Montana,
alden.wright@umontana.edu

Abstract. Beginning with the early days of the genetic algorithm and the schema theorem it has often been argued that the crossover operator is the more important genetic operator. The early Royal Road functions were put forth as an example where crossover would excel, yet mutation based EAs were subsequently shown to experimentally outperform GAs with crossover on these functions. Recently several new Royal Roads have been introduced and proved to require expected polynomial time for GAs with crossover, while needing exponential time to optimize for mutation-only EAs. This paper does the converse, showing proofs that GAs with crossover require exponential optimization time on new Ignoble Trail functions while mutation based EAs optimize them efficiently.

1 Introduction

First proposed by Mitchell et al. [1], the well known Royal Road class of fitness functions were designed to demonstrate the essential nature of the crossover operator in genetic algorithms in optimizing that class of fitness functions. They also showed that for an idealized GA ignoring the effects of hitchhiking, the expected optimization time is $O(2^k \log(n/k))$. Somewhat unexpectedly, follow up experimental studies by Forrest and Mitchell [2] show that some random mutation hill-climbers outperform GAs with crossover on the Royal Road. This prompted the same authors to define an open problem in [3].

– Define a family of functions and prove that genetic algorithms are essentially better than evolutionary algorithms without crossover.

In [4] Jansen and Wegener proved that the expected optimization time of the well known (1+1) EA on the classic Royal Road function is $O(2^k(n/k)\log(n/k))$ where n is the string length, k is the length of sub elements of the string and $1/n$ is the mutation rate. Recently in EA research there have been several fitness functions built to meet this challenge in a rigorous way and are discussed in the next section. The goal of this paper is to do the opposite, to provide a fitness function where EAs with mutation alone are provably better at optimization than GAs with crossover. We are not alone in seeking this result. Very recently Poli et al. have produced a fitness function called OneMix [5] where crossover is shown experimentally to be not helpful.

2 Functions Hard for Mutation Based EAs

In this section we highlight past research on fitness functions hard to optimize with mutation alone, while being much easier with the use of crossover.

The concatenated trap functions of Deb and Goldberg [6] consist of many concatenations of smaller fully deceptive fitness functions. In a fully deceptive function, all points in the space other than the optima give local advice to go in the opposite direction of the optima. By concatenating these functions together, they set up a situation to illustrate where the building block hypothesis [6] shines. Mutation fails to optimize the function, while the crossover operator builds short order sequences of highly fit bits and recombines these sequences to successfully optimize the function.

One explanation for the failure of Royal Road functions to demonstrate the necessity for crossover is that Royal Road functions are both separable and non-deceptive. Watson [7,8] created a hierarchical fitness function called HIFF where sub-blocks are interdependent and non-separable as well as being deceptive to the mutation operator. Non crossover EAs require expected exponential time to optimize the HIFF. Dietzfelbinger et al. [9] asymptotically analyzed a recombinative hill-climber on the HIFF function and showed an expected time complexity of $\Theta(n \log n)$.

Jansen and Wegener [10] showed a unitation fitness function called $\text{JUMP}_{m,n}$ with a deceptive quality. The function contains a false optimum with a neighboring fitness canyon (of size $m < n$) with the true optimum on the other side of the canyon. A steady-state hill-climber that accepts no fitness decreases, like the (μ+1) EA, must simultaneously mutate m bits to cross the canyon. The waiting time for this event is $O(n^m)$, while the waiting time for a steady state GA (with uniform crossover) to optimize $\text{JUMP}_{m,n}$ is $O(n^2 \log n)$ steps.

Jansen and Wegener [4] followed up by introducing Real Royal Road functions where a steady state GA with both uniform and one-point crossover have polynomial expected time. EAs without crossover take expected exponential time to optimize these functions.

Storch and Wegener [11] showed additional Real Royal Roads for both uniform and one-point crossover. Using the (2+1) GA with crossover they proved that these fitness function are optimized in expected polynomial time, while the (2+1) EA will take expected exponential time. The (2+1) EA and GA are redefined in later sections.

One critique of the Real Royal Roads is that they are artificial constructs designed to prove a point. Responding, other researchers have produced works on more natural functions. Fischer and Wegener [12] show a mixed result in the one-dimensional Ising Model where for a correctly chosen λ, the $(1 + \lambda)EA$ performs well compared to typical GAs. They do prove that a specialized GAs do far better than the EA for both one and two point crossover. Sudholt [13] shows that a GA requires polynomial time to optimize another Ising Model, while the EA requires expected exponential time.

Finally, Doerr et al. [14] have an upcoming paper showing that crossover has a provable modest advantage one a real world all pairs shortest path graph problem.

3 Minimal Population Evolutionary Algorithms

These algorithms are instances of steady-state evolutionary algorithms [15] where the population is not fully replaced at each generation. A no-duplicates policy is also in place, forcing a population of distinct strings.

3.1 The Steady-State (2+1) EA

Here we restate the (2+1) EA. It is an instance of the well-known (μ+1) EA, studied among other places in [16].

Algorithm 1. The (2+1) EA

```
1. Initialization: Randomly choose two different individuals $x, y \in \{0,1\}^n$.
2. Search: Produce an individual $z$,
     – with probability 1/2, $z$ is created by mutate($x$),
     – with probability 1/2, $z$ is created by mutate($y$),
3. Selection: Create the new population $\mathscr{P}$.
     – If $z = x$ or $z = y$, then $\mathscr{P} := \{x, y\}$
     – Otherwise, let $a \in \{x, y, z\}$ be randomly chosen among individuals with
       the worst $f$-value. Then $\mathscr{P} := \{x, y, z\} - \{a\}$.
4. Goto Search
```

3.2 The Steady-State (2+1) GA

Here we redefine the simple steady-state GA from [11] that works on a population size of 2, the smallest population size allowing crossover. Note that the usage of equal probability $\frac{1}{3}$ in the search step is arbitrary. The later results hold for any constant probability ϵ where $\epsilon > 0$.

Algorithm 2. The (2+1) GA

```
1. Initialization: Randomly choose two different individuals $x, y \epsilon \{0,1\}^n$.
2. Search: Produce an individual $z$,
     – with probability 1/3, $z$ is created by mutate($x$),
     – with probability 1/3, $z$ is created by mutate($y$),
     – with probability 1/3, $z$ is created by mutate(crossover($x, y$)).
3. Selection: Create the new population $\mathscr{P}$.
     – If $z = x$ or $z = y$, then $\mathscr{P} := \{x, y\}$
     – Otherwise, let $a \epsilon \{x, y, z\}$ be randomly chosen among individuals with
       the worst $f$-value. Then $\mathscr{P} := \{x, y, z :\} - \{a\}$.
4. Goto Search
```

4 Ignoble Trails

We now define a new class of functions called Ignoble Trails. These functions are created for the purpose of rigorously proving that a given mutation based EA outperforms a given crossover based GA on these functions. Like the Real Royal Roads and the HIFF functions, they are somewhat contrived to serve a specific theoretical purpose. We make no claim that real world problems can be mapped to these new functions.

4.1 Ignoble Trails vs. Uniform Crossover

The first function $IT1_n^u(x)$ is a modification of the $R_n^u(x)$ function of [11] for uniform crossover. The symbol u refers to the uniform crossover operator. Most of the details are the same as $R_n^u(x)$ except for the addition of b^{**}. Assume a bit-string length of $n := 6m$, where n and m are even integers. Also note that $\|x\|$ refers to the number of ones in the string, $|x|$ is the length in bits of x, and $H(x, y)$ is the Hamming distance of x and y.

$$IT1_n^u(x) := \begin{cases} 16m & x = b^{**} \\ 15m & x \in T \\ 14m & x = a^* \\ 6m + i & x = a_i \in P_1 \cup P_2 \\ 6m - \|x\| & x \in R := \{0,1\}^n - P - T - \{b^{**}\} \end{cases}$$

The major features of $IT1_n^u(x)$ are as follows. The base fitness of the set R is defined to slope in increasing fitness towards the all zeros string. The path P is a sequence of distinct strings $a_1, ..., a_p$ such that consecutive strings on the path have a Hamming distance of 1. P contains $7m+1$ total points where $a_i = 0^{n-i}1^i$ for $i \leq 6m$, and $a_i = 1^{n-j}0^j$ for $i = 6m + j$. P is segmented into two subpaths P_1 and P_2.

The P_1 subpath is defined as points $(a_0, ..., a_{5m-1})$ and the P_2 subpath is defined as (a_{5m+1}, a_{7m}). The fitness for the total path is $6m + i$, with the single exception that a local optimum is created at point $a^* := a_{5m}$ with fitness $14m$. The other local optimum of P is at the endpoint $a^{**} := a_{7m}$ with fitness value $13m$.

There also exists an area T defined to contain all points $b1^{4m}c$ where the substrings b and c obey $|b| = |c| = m$ and $||b|| = ||c|| = m/2$. In $R_n^u(x)$ T is the target and can be created with high probability with a population of $\{a^*, a^{**}\} := \{0^m1^{5m}, 1^{5m}0^m\}$ via uniform crossover.

Our crucial modification to $R_n^u(x)$ is to add a point b^{**} with fitness greater than the region T. This point is defined as a point with k bits different than a^{**}, or $H(a^{**}, b^{**}) = k$. Here k is defined to be a constant where $n = 6m$ is chosen so that $3 < k < m/4$. We define b^{**} to be $1^m0^k1^{4m-k}0^m$.

4.2 Behavior of the EA and GA on Ignoble Trail 1

Referring to Figure 1, the initial random population of two distinct individuals will begin the process of traveling down R towards the initial point of P, $P_0 := 0^n$. Both algorithms will discover and optimize P unless exceptional luck strikes and $\{T \cup \{b^{**}\}\}$ is discovered first. Since the selection method prohibits duplicate strings, once they are on path P there is a leading point and a trailing

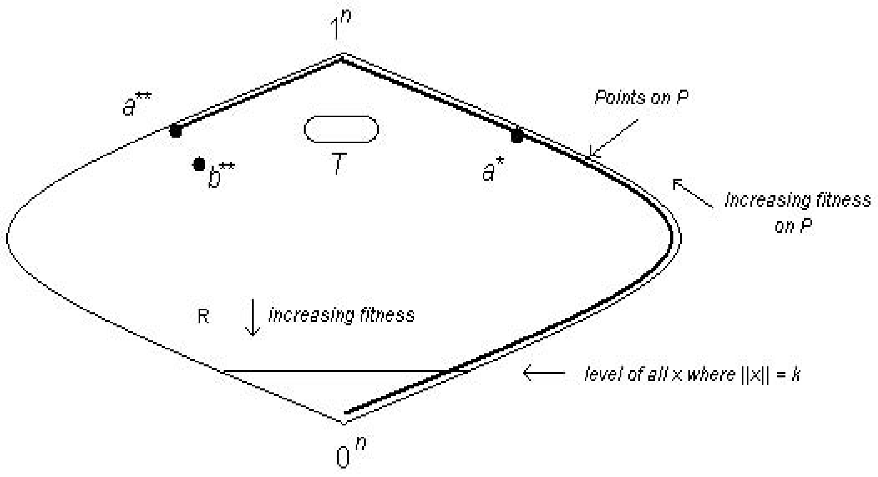


Fig. 1. Illustration of Ignoble Trail 1

point on P. They travel up P until such time as a^* is found [there is a probability $\Theta(1/n)$ a^* is skipped]. If a^* is found, the behavior degenerates to mimic the $(1+1)$ EA as a^* is fixed in the population and the other string is available for continued optimization of P until a^{**} is found.

Once the population becomes $\{a^*, a^{**}\}$ the behavior of the two algorithms diverges. The EA is very unlikely to discover T via mutation, and is likely to find b^{**} in $O(n^k)$ steps. Conversely the GA is very likely to discover T via crossover before it discovers b^{**}. Once the GA has found T, it will accumulate both individuals in T in short order. The expected waiting time to discover b^{**} from T is exponential. Thus we refer to T as the 'trap' rather than the 'target' of $R_n^u(x)$. Note that crossover is of little assistance in discovering b^{**} from either a^{**} or T.

Figure 2 contains a visual representation of the results to follow and the high likelihood optimization phases of both algorithms.

4.3 Time Complexity Results

Note that the next set of proofs take some arguments from [11] or [4]. The addition of b^{**} requires many additional steps to prove rigorous results, there are many more good and bad events to account for above those from [11].

Lemma 1. *The probability that* $(2+1)$ *EA without crossover and the* $(2+1)$ *GA with uniform crossover find a point in* $P_2 \cup T \cup \{b^{**}\}$ *without discovering path* P_1 *within* $O(n^2)$ *steps is at least* $1 - e^{-\Omega(n)}$.

Proof. Recall that k is a constant, and assume that $n = 6m$ is chosen so that $3 < k < m/4$. Let $Q := P_2 \cup T \cup \{b^{**}\}$ and note that all elements of Q have at least $5m - k$ ones. Let R be the set of points not in P with at most $4m$ ones. The probability of initializing a member of the population with more than $4m$

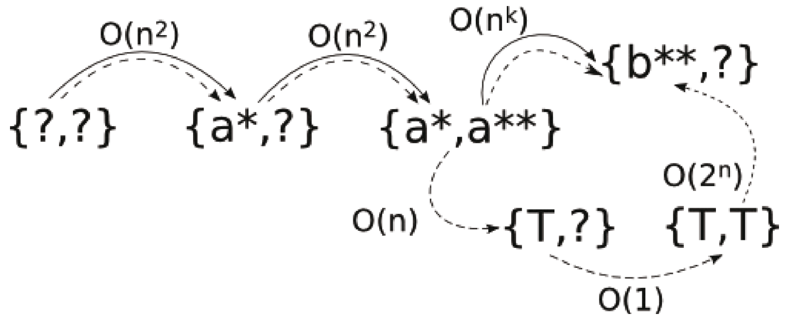


Fig. 2. Diagram of proofs of Lemmas and Theorems for $IT1_n^u(x)$ - Solid lines are events associated with the $(2+1)$ EA, dashed lines are events associated with the $(2+1)$ GA. The labels on each arc refer to the expected waiting time to transition from state to state.

ones is $e^{-\Omega(n)}$ by Chernoff's bound [17]. Since Q is contained in that same set, the same holds for Q. Each point of R has a better Hamming neighbor. The probability of discovering that neighbor via mutation is at least $p = 1/(3en)$. Applying Chernoff bounds, the waiting time for at most $n = 6m$ successful events is $O(n^2)$, and the probability that this waiting time is exceeded is exponentially small. The probability of producing a point in Q from R via mutation is at most $n^{-m+k} = e^{-\Omega(n)}$ by Chernoff's bound. Turning to the crossover operator, the probability of producing a point in Q from two points in R via crossover is $e^{-\Omega(n)}$ by the following argument. Let d be the Hamming distance between the two parent strings r_1 and r_2. Let $s = ||r_1 \wedge r_2||$, thus the expected number of ones is $s + d/2$. Unless $d > m - k$, the child string can not have at least $5m - k$ ones. Applying Chernoff's bound on the differing bits of the parents, $r_1 \oplus r_2$, the probability to create at least $d/2 + m - k$ ones is $e^{-\Omega(n)}$. As for the joint operator, the probability of producing a point in Q from two points in R via crossover and mutation is $e^{-\Omega(n)}$ as follows. Either crossover produces a point with at least $9m/2 - k$ ones or it doesn't. In the first case, the probability that crossover produces a point with at least $9m/2 - k$ ones is $e^{-\Omega(n)}$ by the Chernoff bounds on the the bits differing in the parents. In the other case, mutation must go from a point with less than $9m/2 - k$ ones to a point with at least $5m$ ones, and the probability that this happens $n^{-m/2+k} = e^{-\Omega(n)}$. Applying the union bound, we see that the total failure probability is $e^{-\Omega(n)}$. □

Lemma 2. *The* $(2+1)$ *EA will optimize* P *and find* $\{a^{**} \cup b^{**}\}$ *in* $O(n^2)$ *steps with probability* $1 - 2^{-\Omega(n)}$*. The* $(2+1)$ *EA will discover a point in* T *from* P *with probability* $2^{-\Omega(n)}$*.*

Proof. Beginning from Lemma 1, we assume the population contains a point in P_1. Each point on the path P has a better fitness Hamming neighbor. The probability of discovering that neighbor via mutation is at least $p = 1/(3en)$. Inverting and substituting we get a waiting time of at most $7m$ (the length of P) successful events of probability p. Applying Chernoff's bound we get the first result above. As for the second result, by the definitions of P and T, the Hamming distance between them is at least $m/2$. The mutation hitting probability is $(1/n)^{m/2}(1-1/n)^{n-m/2}$. However, there are $\binom{m}{m/2}^2$ points in T, so the probability of hitting T is increased by this amount. Bounding the number of points in T via a standard binomial coefficient inequality[1], we get $\binom{m}{m/2}^2 \leq (2e)^{m/2}$. Thus we bound the probability of hitting T from $p_i \in P$ by $(1/n)^{m/2}(1-1/n)^{n-m/2}(2e)^{m/2} \leq (2e/n)^{m/2} < 2^{-\Omega(n)}$. □

Theorem 3. *The (2+1) EA without crossover will optimize the $IT1_n^u(x)$ function in expected $O(n^k)$ steps and within $O(n^k \ln k)$ steps with probability $1-O(1/n)$.*

Proof. Referring to Lemma 2, the next step is to establish the expected waiting time to discover b^{**} from a population of $\{a^{**}, p_i \in P\}$. The Hamming distance between a^{**} and b^{**} is defined to be constant k where $n = 6m$ is chosen so that $3 < k < m/4$. Thus the probability of mutating from a^{**} to b^{**} in one step is $p = (1/n)^k(1-1/n)^{n-k}$. This is bounded below by $1/(en^k)$, resulting in an expected waiting time that is bounded above by $en^k = \Theta(n^k)$. Note that this is the best case possibility of finding b^{**} from any point on P as the Hamming distance for all points in P is $H(p_i \in P, b^{**}) \geq k$. Applying Chernoff bounds, the probability of finding b^{**} within $en^k \ln k$ steps is $1-O(1/n)$. From Lemma 2 we know that the probability of finding T from any point in P is exponentially small. Thus the probability of finding T before finding b^{**} is also exponentially small. □

Lemma 4. *The $(2+1)$ GA with uniform crossover will discover a point in $P_2 \cup T \cup \{a^*\}$ in $O(n^2)$ steps with probability $1-2^{-\Omega(n)}$. The probability of the $(2+1)$ GA with uniform crossover finding $\{b^{**}\}$ while searching for $P_2 \cup T \cup \{a^*\}$ is $2^{-\Omega(n)}$.*

Proof. Lemma 2 of [11] proves the first part of the result. For the second result, note that b^{**} contains $5m-k$ ones. Recall that k is a constant, and assume that $n = 6m$ is chosen so that $3 < k < m/4$. We have already shown that as long as the points in the population contain at least $4m$ ones, the probability of finding b^{**} is exponentially small. The remaining possibility is mutating to b^{**} from a point in the population $p_i \in \{P_1 - a^*\}$ where $a_{4m} < p_i < a_{5m}$. It is easy to see that it is exponentially unlikely that the other point of the population is not in $\{a_i \in P_1\ i \geq m\}$. The minimum Hamming distance between a point of the population and b^{**} is $2m-k$, so the probability of finding b^{**} by mutation is at most $2^{-\Omega(n)}$. Turning to the crossover operator, recall that $b^{**} = 1^m 0^k 1^{4m-k} 0^m$. Both members of the population are of the form $0^{n-i}1^i$ for $m \leq i < 5m$ so both points have 1s in the last m positions. Thus, it is impossible to cross the two points in the population to produce a point with Hamming distance less than m from b^{**}. □

[1] $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$.

Proposition 5. *With probability $O(1/n)$, the $(2+1)$ GA will find a point in $P_2 \cup T \cup \{b^{**}\}$ before finding a^*.*

Proof. The proof of Theorem 4 of [11] shows this result without reference to b^{**}. The Hamming distance from P_1 to b^{**} is exponential, and thus does not change the result. □

Lemma 6. *If the population contains a^*, the $(2+1)$ GA will find a point in T in $O(n^2)$ steps with probability $1 - O(1/n^k)$.*

Proof. Lemma 3 of [11] shows the result with probability $1 - 2^{-\Omega(n)}$. We must consider the possibility that b^{**} is found before T. To start, we consider the possibility of finding b^{**} by crossover plus mutation from a population of a^* and any other point $a_i \in P$. For $0 \leq i \leq 5m$ this is exponentially unlikely via the argument given in the proof of Lemma 4. For $5m < i \leq 6m$ this results in crossover on a^* and a_i setting the last m positions to 1. Yet b^{**} has zeros in these positions, so subsequent mutation must flip at least m bits. Finally, if the other point is $a_{6m+j} = 1^{n-j}0^j$ for $0 < j \leq m$, then a^* and a_{6m+j} agree in $k + m - j$ bits different from the corresponding bits of b^{**}. Thus crossover and subsequent mutation of at least those $k + m - j$ bits is required, giving a probability of discovering b^{**} bounded above by $O(1/n^{k+m-j})$. As long as a_i is not a^{**}, a better point on P will be discovered with probability $1/(3en)$. From this and the bounds derived above, we can see that either a^{**} or a point of T will be found with probability $1 - O(1/n^k)$.

Now assume the population $\{a^*, a^{**}\}$. The one-step probability of finding b^{**} by either mutation or crossover followed by mutation is $p = O(1/n^k)$ whereas the one-step probability of discovering T was shown to be $q = \Theta(1/n)$ in Lemma 3 of [11] by an application of Sterling's formula. There is a sequence of independent trials until one or the other of these outcomes happens. A probability argument[2] shows that the probability of finding b^{**} over all trials is $p/(p+q) = O(1/n^k)/(O(1/n^k) + O(1/n)) = O(1/n^k)/O(1/n) = O(1/n^{k-1})$. □

Lemma 7. *The expected waiting time to hit b^{**} from a population $\{t_i \in T, t_j \in T\}$ is exponential for the (2+1) GA with uniform crossover.*

Proof. It is possible for a crossover plus mutation operation to get b^{**} from two elements of T. Remember that $b^{**} := 1^m 0^k 1^{4m-k} 0^m$. If the two population elements of T are binary complements of each other in the b and c regions, and if the crossover mask is chosen correctly, crossover could get the first and last m bits of the child to match b^{**}. Then mutation would need to get the k bits of the middle 1^{4m} bits to match b^{**}. The probability of getting the correct crossover mask is 2^{-2m}. Thus the probability of getting the correct mask and the correct mutation is bounded above by $O(2^{-2m})$.

Another possibility would be for crossover to get all but $0 \leq j \leq 2m$ of the first and last m bits correct. These correspond to the substrings b and c from the

[2] Given that either event A or B will eventually happen, let $p := Pr[A]$, $q := Pr[B]$ and $r := 1-p-q$. The probability that A eventually happens is $p/(1-r) = p/(p+q)$.

definition of T, $b1^{4m}c$ where b and c contain exactly half 1s. It is not necessary for these j bits of the crossover mask to be correct, thus the probability of choosing the correct crossover mask is 2^{-2m+j}. Following crossover, mutation must correct $k+j$ bits, with probability $(1/n)^{k+j}(1-1/n)^{n-k-j} \leq (1/n)^{k+j}$. Consequently, the probability of getting the crossover mask right and the correct mutation is $\leq (1/n)^{k+j}(1/2)^{2m-j} \leq (1/2)^{2m+j}$ which is exponentially small. □

Theorem 8. *The (2+1) GA with uniform crossover will need exponential time steps to optimize $IT1^u_n(x)$ with probability $1 - O(1/n)$.*

Proof. Beginning from Prop. 5 and Lemma 6 above, assume the population contains a point in T. By the selection method of the GA, once a member of T exists in the population we should only have to wait constant time $O(1)$ for both members of the population to be in T. Once the GA contains two members of T, probability of crossover plus mutation or mutation alone discovering b^{**} is exponentially unlikely by Lemma 7. Of the various bad events, the probability from Prop. 5 of skipping a^* is maximal at $O(1/n)$. □

5 Conclusions

We believe we have shown for the first time a proven example of a situation where a crossover based GA is expected to be exponentially outperformed by an EA without the crossover operator. Future work will expand upon this result with empirical studies and extensions to cover 1-point crossover. In addition it is believed that examples can be created for large population EA/GAs showing this exponential performance difference. An open problem would be to follow up on [14] and produce a reasonable graph problem that where the GA is outperformed by an EA.

Acknowledgments

The authors would like to thank Thomas Jansen, Ingo Wegener, Tomas Gedeon and the anonymous reviewers for their encouragement and advice. We would also like to thank the participants and organizers of the Theory of Evolutionary Algorithms conference at Schloss Dagstuhl International Conference and Research Center for Computer Science, where the seeds of this research were germinated.

References

1. Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and GA performance. In: Varela, F.J., Bourgine, P. (eds.) Proceedings of the First European Conference on Artificial Life. Toward a Practice of Autonomous Systems, pp. 243–254. MIT Press, Cambridge (1992)
2. Forrest, S., Mitchell, M.: Relative building-block fitness and the building-block hypothesis. In: Whitley, L.D. (ed.) Foundations of genetic algorithms 2, pp. 109–126. Morgan Kaufmann, San Mateo (1993)

3. Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) NIPS, pp. 51–58. Morgan Kaufmann, San Francisco (1993)
4. Jansen, T., Wegener, I.: Real royal road functions–where crossover provably is essential. Discrete Applied Mathematics 149(1-3), 111–125 (2005)
5. Poli, R., Wright, A.H., McPhee, N., Langdon, W.: Emergent behaviour, population-based search and low-pass filtering. In: Proceedings of the Congress on Evolutionary Computation (CEC) 2006, pp. 88–95. IEEE Press, Los Alamitos (2006)
6. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) FOGA, pp. 93–108. Morgan Kaufmann, San Francisco (1992)
7. Watson, R.A.: Analysis of recombinative algorithms on a non-separable building-block problem. In: Foundations of Genetic Algorithms 6, pp. 69–89. Morgan Kaufmann, San Francisco (2001)
8. Watson, R.A., Pollack, J.B.: Hierarchically consistent test problems for genetic algorithms. In: Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.) Proceedings of the Congress on Evolutionary Computation, Mayflower Hotel, Washington D.C., USA, vol. 2, pp. 6–9. IEEE Press, Los Alamitos (1999)
9. Dietzfelbinger, M., Naudts, B., Hoyweghen, C.V., Wegener, I.: The analysis of a recombinative hill-climber on h-iff. IEEE Trans. Evolutionary Computation 7(5), 417–423 (2003)
10. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms - a proof that crossover really can help. Algorithmica 34(1), 47–66 (2002)
11. Storch, T., Wegener, I.: Real royal road functions for constant population size. Theor. Comput. Sci. 320(1), 123–134 (2004)
12. Fischer, S., Wegener, I.: The one-dimensional ising model: Mutation versus recombination. Theor. Comput. Sci. 344(2-3), 208–225 (2005)
13. Sudholt, D.: Crossover is provably essential for the ising model on trees. In: GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1161–1167. ACM, New York (2005)
14. Doerr, B., Happ, E., Klein, C.: Crossover is provably useful in evolutionary computation. In: GECCO 2008: Proceedings of the 2008 conference on Genetic and evolutionary computation (to appear, 2008)
15. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
16. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Dr. Kovac, Hamburg (1997)
17. Motwani, R., Raghavan, P.: Randomized algorithms. Cambridge University Press, New York (1995)

Lower Bounds for Evolution Strategies Using VC-Dimension

Olivier Teytaud[1] and Hervé Fournier[2]

[1] TAO (Inria), LRI, UMR 8623 (CNRS - Univ. Paris-Sud), Bât 490
Univ. Paris-Sud 91405 Orsay, France
`teytaud@lri.fr`
[2] Laboratoire PRiSM
CNRS UMR 8144 and Univ. Versailles St-Quentin en Yvelines
45 av. des États-Unis, 78035 Versailles, France
`herve.fournier@prism.uvsq.fr`

Abstract. We derive lower bounds for comparison-based or selection-based algorithms, improving existing results in the continuous setting, and extending them to non-trivial results in the discrete case. This is achieved by considering the VC-dimension of the level sets of the fitness functions; results are then obtained through the use of Sauer's lemma. In the special case of optimization of the sphere function, improved lower bounds are obtained by bounding the possible number of sign conditions realized by some systems of equations.

Keywords: Evolution Strategies, Convergence ratio, VC-dimension, Sign conditions.

1 Introduction

Evolution strategies (ES), defined by Rechenberg [15], are a family of optimization algorithms with nice robustness properties. Most ES use only comparisons between fitness values and not the fitness values themselves. This fact has been used in [18] in order to provide lower bounds that match some upper bounds known for evolutionary algorithms [8,2,16]. The optimality of this comparison-based principle for some robustness criterion was shown in [10] (see also [3,20,4]). In [18] is provided a new tool for proving lower bounds for evolutionary algorithms, but, as pointed out by the authors, some bounds are not tight and in particular: (i) the discrete case provides essentially trivial results; (ii) the bounds for the (μ, λ)-ES are far too large. In this work, we propose improved lower bounds for evolution strategies of type $(\mu \overset{+}{,} \lambda)$-ES (i.e. upper bounds on the convergence ratios of these algorithms) in terms of the VC-dimension of level sets of the fitness functions. In the special case of optimization of the sphere function, improved upper bound on the convergence ratio of evolution strategies are presented; they are obtained by bounding the number of sign conditions realized by a system of equations. The paper is organized as follows. Basic definitions and terminology of evolution strategies we consider are described in Section 2.

Lower bounds on $(\mu \overset{+}{,} \lambda)$-ES based on the branching factor, obtained in [18], are recalled in Section 3. Improved lower bounds on $(\mu \overset{+}{,} \lambda)$-ES in terms of the VC-dimension are presented in Section 4. At last, some questions are raised in Section 5.

Notations. In all the paper, $\log(x)$ denotes the logarithm with basis 2, i.e. $\log(2) = 1$. The set of integers $\{1, 2, \ldots, n\}$ is denoted by $[[1, n]]$.

2 Evolution Strategies of Type $(\mu \overset{+}{,} \lambda)$

We define in this section $(\mu \overset{+}{,} \lambda)$-algorithms – we refer to Beyer and Schwefel [6] for a comprehensive introduction to evolution strategies. The aim of a $(\mu \overset{+}{,} \lambda)$-algorithm is to find the minimum of a function f (called the fitness function) defined over a domain D. This algorithm cannot evaluate the function f but has to work only with comparisons: given two points x and y, the algorithm has access to a black-box telling whether $f(x) < f(y)$, $f(x) = f(y)$ or $f(x) > f(y)$. Of course such an algorithm is not required to work for one fitness function but for a whole family of fitness functions. In the following we denote by $\mathcal{F}$ the set of fitness functions we consider. In the rest of the paper, we assume we never have a case of equality $f(x) = f(y)$ among the generated points. Let λ

Algorithm 1. SB-(μ, λ)-ES (resp. SB-$(\mu+\lambda)$-ES), i.e. evolution strategies based on selection, working on a fitness function f. The real number ω is a random seed, uniform in $[0, 1]$. We do not specify the generation of the offspring, because we work on the whole family of algorithms matching this framework.

```
Initialize $I_0 \in \mathcal{I}$, $S_{-1} = \emptyset$ and $n = 0$
while true do
    Generate an offspring $O_n$ of $\lambda$ distinct points: $O_n = \mathrm{generate}(I_n, \omega)$.
    Selection: Use the fitness $f$ in order to partition $O_n$ (resp. $O_n \cup S_{n-1}$) in two sets
    $S_n$ of cardinal $\min(\mu, \mathrm{Card}(O_n))$ and $R_n$ such that
            $x \in S_n$ and $y \in R_n \Rightarrow f(x) < f(y)$.
    We denote this by $S_n = \mathrm{select}(O_n, f)$ (resp. $S_n = \mathrm{select}(O_n \cup S_{n-1}, f)$).
    Update the internal state:
        $I_{n+1} = \mathrm{update}(I_n, f, O_n) = \mathrm{selectionUpdate}(I_n, S_n, R_n) \in \mathcal{I}$.
    $x^{(f)}_{\omega,n+1} = \mathrm{proposal}(I_n)$
    $n = n + 1$
end while
```

and μ bee two integers (subject to $\mu \leqslant \lambda$ in the (μ, λ) case). A SB-$(\mu \overset{+}{,} \lambda)$-ES (Selection Based $(\mu \overset{+}{,} \lambda)$-ES) is an algorithm working as follows. There is a set $\mathcal{I}$ of internal states and an initial state I_0. At each iteration, the algorithm follows these three successive steps. First generate a set of λ points, called the *offspring*. Then select only the μ best ones, i.e. the μ points with lowest fitness values;

Algorithm 2. (μ, λ)-ES (resp. $(\mu + \lambda)$-ES) based on full ranking, working on a fitness function f. The real number ω is a random seed, uniform in $[0, 1]$. Compared to Algorithm 1, S_n is now a vector of points, ordered with respect to their fitness values. This family of algorithms is more general than Algorithm 1, as we can use all the ranking information.

```
**Initialize** $I_0 \in \mathcal{I}$, $S_{-1} = \emptyset$ and $n = 0$
**while** true **do**
    **Generate** an offspring $O_n$ of $\lambda$ distinct points: $O_n = \text{generate}(I_n, \omega)$.
    **Selection with ranking:** Use the fitness $f$ in order to partition $O_n$ (resp. $O_n \cup S_{n-1}$) in a vector $S_n = (x'_1, \ldots, x'_{c_n})$ of cardinal $c_n = \min(\mu, \text{Card}(O_n))$ (resp. $c_n = \min(\mu, \text{Card}(O_n \cup S_{n-1}))$) and a set $R_n$ such that
        $\forall i \in [[1, c_n]], \forall y \in R_n, f(x'_i) < f(y),$
        and $\forall i \in [[1, c_n - 1]], f(x'_i) < f(x'_{i+1})$.
    We denote this by $S_n = \text{select}(O_n, f)$ (resp. $S_n = \text{select}(O_n \cup S_{n-1}, f)$).
    **Update** the internal state:
        $I_{n+1} = \text{update}(I_n, f, O_n) = \text{fullRankUpdate}(I_n, S_n, R_n) \in \mathcal{I}$.
    $x^{(f)}_{\omega,n+1} = \text{proposal}(I_n)$
    $n = n + 1$
**end while**
```

in the case of a SB-(μ, λ)-ES, points generated at previous stages are forgotten and this selection is performed only among the offspring, while an algorithm of type SB-$(\mu + \lambda)$-ES selects the μ best points among the offspring *and* the points selected at the previous step (hence these μ selected points are always the μ points with lowest fitness values found so far). At last the internal state is updated. General outlines of SB-(μ, λ)-algorithms (resp. SB-$(\mu + \lambda)$-algorithms) are summarized in Algorithm 1.

Algorithms with the "+" are usually termed *elitist*; this means that we always keep the best individuals. Algorithms with the "," are termed *non-elitist.* Elitist strategies are usually faster on easy fitness functions, but less robust; therefore, non-elitist strategies are usually prefered.

At last we would like to explain a generalization of SB-$(\mu \overset{+}{,} \lambda)$-ES, called $(\mu \overset{+}{,} \lambda)$-ES. Instead of just giving the best μ points (i.e. the μ points with the lowest fitness values), we can consider a selection procedure which returns the best μ points *ordered with respect to their fitness.* More precisely, given the points $(y_1, \ldots, y_p)$ (O_n in the case of (μ, λ)-ES or $O_n \cup S_{n-1}$ in the case of $(\mu + \lambda)$-ES), it returns μ distinct indices $(i_1, \ldots, i_\mu)$ such that $f(y_{i_1}) < \ldots < f(y_{i_\mu})$ and for all $j \notin \{i_1, \ldots, i_\mu\}$, $f(y_{i_\mu}) < f(y_j)$. We call *full ranking* this kind of "selection" [4,3,20]. The outline of these algorithms is summarized in Algorithm 2.

Note that both Algorithms 1 and 2 define a class of algorithms: in order to obtain an algorithm, one has to specify how generation of points is done, what is the set of internal states as well as the update function. We assume that all functions involved in these algorithms are measurable. A usual case is retrieved when the offspring is randomly and independently drawn according to a Gaussian distribution, with parameters (mean, variance and covariances) depending on the internal state of the algorithm.

3 Branching Factor and Convergence Ratio

We consider a (possibly discrete) domain $D \subset \mathbb{R}^d$ and a norm $\|\cdot\|$ on $\mathbb{R}^d$. For $\varepsilon > 0$, we define $N(\varepsilon)$ to be the maximum integer n such that there exist n distinct points $x_1, \ldots, x_n \in D$ with $\|x_i - x_j\| \geqslant 2\varepsilon$ for all $i \neq j$. If each function $f \in \mathcal{F}$ has one and only one optimum f^*, for any given optimization algorithm as in Algorithm 2, and for $\varepsilon > 0$ and $\delta > 0$, we let $n_{\varepsilon,\delta}$ be the minimum number n of iterations such that with probability at least $1-\delta$, an optimum is found at the n-th iteration within distance ε. I.e. $n_{\varepsilon,\delta}$ is minimal such that for all $n \geqslant n_{\varepsilon,\delta}$ and for all $f \in \mathcal{F}$,

$$\mathbb{P}_{w\in[0,1]}[\|x^{(f)}_{\omega,n} - f^*\| < \varepsilon] \geqslant 1-\delta.$$

For an algorithm of type $(\mu \overset{+}{,} \lambda)$-ES working over a set $\mathcal{F}$ of fitness functions, we define the *branching factor* of any algorithm as in Algorithm 2 as

$$K = \sup_{I\in\mathcal{I},O} \text{Card}\{\text{update}(I,f,O) \mid f \in \mathcal{F}\}.$$

Notice that in the case of selection based algorithms (any algorithm fitting Algorithm 1), we have

$$K \leqslant \sup_{O} \text{Card}\{\text{select}(O,f) \mid f \in \mathcal{F}\}$$

where the supremum holds for: (i) O any set of λ points in the case of SB-(μ,λ)-ES; (ii) O any set of $\lambda+\mu$ points in the case of SB-$(\mu+\lambda)$-ES. A similar remark holds in the case of full ranking $(\mu \overset{+}{,} \lambda)$-ES, except that a bound on K is given by the possible number of choices of selected points together with their order (with respect to their fitness values). Let us recall the following result from Teytaud and Gelly [18] (restricted here to our purpose) relating the convergence ratio and the branching factor of a $(\mu \overset{+}{,} \lambda)$-ES.

Theorem 1 (Lower bound on the convergence ratio of $(\mu \overset{+}{,} \lambda)$-ES.). *Consider a (μ,λ)-ES or $(\mu+\lambda)$-ES as in Algorithm 2. Consider a set $\mathcal{F}$ of possible fitness functions on domain D, i.e. $\mathcal{F} \subset \mathbb{R}^D$, such that any fitness function $f \in \mathcal{F}$ has only one min-argument f^*, and such that $\{f^* \mid f \in \mathcal{F}\} = D$. Let $\varepsilon > 0$ and $\delta \in]0,1[$. Let $L_n(\omega)$ be the number of different paths (when the function f runs over $\mathcal{F}$) followed by the algorithm on the random seed ω after n steps of computation; then*

$$\mathbb{E}_{\omega\in[0,1]}[L_{n_{\varepsilon,\delta}}(\omega)] \geqslant (1-\delta)N(\varepsilon).$$

In particular, if K denotes the branching factor of the algorithm, then

$$n_{\varepsilon,\delta} \geqslant \left\lceil \frac{\log(1-\delta)}{\log(K)} + \frac{\log(N(\varepsilon))}{\log(K)} \right\rceil.$$

We can define the convergence ratio for both discrete and continuous domains thanks to the following unified definitions. We want a definition of convergence ratio which matches bounds of the form $O(1/d)$ established in [13]; therefore, we define the *convergence ratio* of an algorithm for precision ε as

$$\mathrm{CR}_\varepsilon = \frac{\log N(\varepsilon)}{d n_{\varepsilon,\frac{1}{2}}}.$$

We also define the *normalized convergence ratio* (normalized by the number of individuals generated per epoch) by

$$\mathrm{NCR}_\varepsilon = \frac{\log N(\varepsilon)}{d\lambda n_{\varepsilon,\frac{1}{2}}}.$$

The ratio CR_ε is relevant in the parallel setting (i.e. it is the convergence ratio when working on a parallel computer, with parallel evaluation of the offspring), while NCR_ε is relevant in the sequential setting, i.e. when individuals are evaluated sequentially.

Theorem 1 can be reformulated with these unified definitions of convergence ratios as follows. Consider a $(\mu \overset{+}{,} \lambda)$-ES satisfying the hypothesis of Theorem 1. Let $\alpha(\varepsilon) = 1/(1 - 1/N(\varepsilon))$. Then

$$\mathrm{CR}_\varepsilon \leqslant \frac{\log K}{d} \cdot \alpha(\varepsilon) \text{ and } \mathrm{NCR}_\varepsilon \leqslant \frac{\log K}{d\lambda} \cdot \alpha(\varepsilon). \quad (1)$$

4 Sauer's Lemma and VC-Dimension

Teytaud and Gelly [18] applied the bounds obtained in Section 3 in the following way: the number of subsets of size μ of a set of λ points, is at most $\binom{\lambda}{\mu} \leqslant \binom{\lambda}{\lfloor \lambda/2 \rfloor} \leqslant (2^\lambda/\sqrt{2\pi\lambda})$ – see e.g. [7, p587] or [9] for these inequalities. This surely holds, but it is a worst case on possible selections: if the fitness funtions are "nice", many of these subsets cannot be realized. This is precisely quantified by Sauer's lemma in the theory of VC-dimension. In this section, we show how this allows to obtain more precise lower bounds on the convergence ratio of $(\mu \overset{+}{,} \lambda)$-ES.

Given a function f defined over D and $r > 0$, let $O_{f,r} = \{x \in D \mid f(x) < r\}$. We define the *level sets* $L_\mathcal{F}$ of a set $\mathcal{F}$ of functions defined over the domain D as

$$L_\mathcal{F} = \{O_{f,r} \mid f \in \mathcal{F}, r > 0\}.$$

We now briefly recall the definition of VC-dimension and Sauer's lemma [19,17] – our presentation is based on [14]. A set system on a set A is a family $\mathcal{S}$ of subsets of A. For $B \subseteq A$, we define the restriction of $\mathcal{S}$ to B as $\mathcal{S}|_B = \{S \cap B \mid S \in \mathcal{S}\}$. The VC-dimension of the set system $\mathcal{S}$ defined over A is defined as $\sup\{|B| \mid \mathcal{S}|_B = 2^B\}$ where 2^B denotes the powerset of B; in other words, it is the size of the largest subset B of A such that any subset of B can be obtained by intersecting B with an element of $\mathcal{S}$. Given a set system $\mathcal{S}$ over A, the shatter function $\pi_{\mathcal{S}}$ is defined by $\pi_{\mathcal{S}}(m) = \max\{|\mathcal{S}|_B| \mid B \subseteq A,\ |B| = m\}$; thus $\pi_{\mathcal{S}}(m)$ is the maximum number of different subsets of A which can be obtained by intersecting a single subset of size m of A with all elements of $\mathcal{S}$. We next recall Sauer's lemma which gives an upper bound on $\pi_{\mathcal{S}}$ in terms of the VC-dimension of $\mathcal{S}$.

Lemma 1 (Sauer's lemma). *For any set system $\mathcal{S}$ of VC-dimension d, then for all integer m, it holds that $\pi_{\mathcal{S}}(m) \leqslant \sum_{i=0}^{d} \binom{m}{i}$.*

At last, let us recall the following classical bound [7] which is valid whenever $d \geqslant 3$:

$$\sum_{i=0}^{d} \binom{m}{i} \leqslant \min\{m^d, 2^m\}. \tag{2}$$

Note that the trivial bound 2^m is tight when $m \leqslant d$. The interesting case happens when m is large with respect to the VC-dimension d: the bound becomes polynomial in m in this case. This element is central for the difference between the results in this paper and results in [18].

In the rest of the paper, we assume the VC-dimension of considered set systems is always at least 3 (however, the case of VC-dimension smaller than 3 can be handled in a similar way; the bound above has to be replaced with $\sum_{i=0}^{d} \binom{m}{i} \leqslant m^d + 1$).

4.1 Non-elitist Strategies

We first give an upper bound on the branching factor of a SB-(μ, λ)-ES in terms of the VC-dimension of level sets.

Lemma 2. *Consider a SB-(μ, λ)-ES as described in Algorithm 1. Let $V \geqslant 3$ be the VC-dimension of the level sets of the family $\mathcal{F}$ of fitness functions under consideration. Then the branching factor of this algorithm satisfies $K \leqslant \lambda^V$.*

Proof. Given a set of λ points $P = \{x_1, \ldots, x_\lambda\}$ in the domain D, and $f \in \mathcal{F}$, let us define $M_f(P)$ to be the subset Q of size μ of P correponding to the μ points of P with lowest fitness values with respect to f. Note that the branching factor satisfies

$$K \leqslant \max_{P \subset D,\ |P| = \lambda} |\{M_f(P) \mid f \in \mathcal{F}\}|.$$

Now remark that for any P, the set Q of the μ points of P with lowest value (with respect to the fitness function f) can be separated from $P \setminus Q$ by an element from the level sets: in other words, there exists $O \in L_{\mathcal{F}}$ such that $O \cap P = Q$. It follows that

$$|\{M_f(P) \mid f \in \mathcal{F}\}| \leqslant \pi_{L_{\mathcal{F}}}(\lambda).$$

If the VC-dimension of $L_{\mathcal{F}}$ is at most V, it follows from Sauer's lemma and the bound given in Equation 2 that $\pi_{L_{\mathcal{F}}}(\lambda) \leqslant \lambda^V$. Thus $K \leqslant \lambda^V$. □

Theorem 2 (SB-(μ,λ)-ES). *Consider a SB-(μ,λ)-ES (Algorithm 1) in a domain $D \subset \mathbb{R}^d$, such that $D = \{f^* \mid f \in \mathcal{F}\}$. Let $V \geqslant 3$ be the VC-dimension of the level sets of $\mathcal{F}$. The convergence ratio of this algorithm satisfies*

$$\mathrm{CR}_\varepsilon \leqslant \frac{V \log \lambda}{d} \cdot \alpha(\varepsilon),$$

where $\alpha(\varepsilon) = 1/(1 - 1/N(\varepsilon))$.

Proof. The result easily follows from the upper bound on the branching factor given in Lemma 2, and from Theorem 1 as stated in Equation 1. □

4.2 Non-elitist Strategies with Full Ranking

This subsection deals with algorithms of type full ranking (μ,λ)-ES. It is organized as follows:

- First we study to which extent lower bounds obtained for SB-(μ,λ)-ES are modified when we use the full ranking information and not only selection information (i.e. we move from Algorithm 1 to Algorithm 2);
- Although the bounds obtained in the general case do not forbid a linear speed-up in λ, we show that the speed-up is asymptotically at most logarithmic in the special case of the sphere function;
- At last, for the sphere function again, we remark that a convergence ratio $\mathrm{CR}_\varepsilon = \Theta(1)$ can be reached in the case $\lambda = 2d$; this is to be compared to the best convergence ratio $\mathrm{CR}_\varepsilon = \Theta(1/d)$ we are aware of for $\lambda = O(1)$.

Keeping the full ranking information. Consider the case of Algorithm 2 instead of Algorithm 1; we have a wider family of algorithms as we can use all the ranking information. There are evolutionary algorithms which use the full ranking information of the selected points and not only selection; for example, roulette-wheel with rank-based fitness assignment (stochastic sampling [4], rank-based fitness assignment [3,20]), weighted recombination [11,1] or BREDA [10]. In this case, an upper bound on the number of possible outcomes of the selection step (including the ranking of children) is obtained by multiplying by $\mu!$ the number of possible outcomes in the case of selection only. This gives $\mathrm{CR}_\varepsilon \leqslant \frac{V\log(\lambda)+\mu\log\mu}{d} \cdot \alpha(\varepsilon)$. However, we can say better in the case where μ is large with respect to the VC-dimension V of the level sets of the fitness functions. (Proof of the following theorem is omitted due to space limitations.)

Theorem 3 (Full ranking (μ,λ)-ES). *Consider a (μ,λ)-ES (Algorithm 2) in a domain $D \subset \mathbb{R}^d$, such that $D = \{f^* \mid f \in \mathcal{F}\}$. Let $V \geqslant 3$ be the VC-dimension of the level sets of $\mathcal{F}$. The convergence ratio of this algorithm satisfies*

$$\mathrm{CR}_\varepsilon \leqslant \frac{V\,(\log \lambda + 4\mu)}{d} \cdot \alpha(\varepsilon),$$

where $\alpha(\varepsilon) = 1/(1 - 1/N(\varepsilon))$.

The case of the sphere function: complexity bounds for λ large. For the sphere function and the Euclidean norm, we next give an upper bound on the convergence ratio of a selection-based algorithm using full ranking.

Proposition 1. *Let $d \geqslant 3$. Consider a (μ, λ)-ES, as in Algorithm 2, optimizing the sphere function in a domain $D \subset \mathbb{R}^d$. Then $\mathrm{CR}_\varepsilon \leqslant 2\log(\lambda)\cdot\alpha(\varepsilon)$, where $\alpha(\varepsilon) = 1/(1 - 1/N(\varepsilon))$.*

Proof. Given two distinct points p and q in $\mathbb{R}^d$, we denote by $H_{p,q}$ be the mediator hyperplane of p and q, i.e. $H_{p,q} = \{x \in \mathbb{R}^d \mid \|x - p\| = \|x - q\|\}$.

At each iteration of the algorithm, an offspring of λ points $\{x_1, \ldots, x_\lambda\}$ is generated and the algorithm receives the sequence of indices of the μ points with lowest fitness values, ordered with respect to their fitness values. Obviously the branching factor is maximal when $\mu = \lambda$, i.e. when the algorithm is given the full ordering of points with respect to their fitness values. This information corresponds to giving the sign $s_{i,j}$ of $f(x_i) - f(x_j)$ for each $1 \leqslant i < j \leqslant \lambda$; this sign is positive or negative since we assumed equality never occurs. The number of possible sign vectors $s = (s_{i,j})_{1\leqslant i<j\leqslant\lambda}$ is exactly the number of cells of the arrangement of hyperplanes $\{H_{x_i,x_j} \mid 1 \leqslant i < j \leqslant \lambda\}$ in $\mathbb{R}^d$. But it is known that n hyperplanes in $\mathbb{R}^d$ define at most n^d cells – see chapter 6 of [14]. Since there are $\binom{\lambda}{2} \leqslant \lambda^2/2$ hyperplanes here, we obtain $K \leqslant \left(\lambda^2/2\right)^d$. Applying Equation 1 yields the announced bound on the convergence ratio. □

When ε tends towards 0 and as $N(\varepsilon) \to \infty$, this gives $\mathrm{CR}_\varepsilon \leqslant 2\log\lambda$; this shows that the upper bound given by Theorem 3 cannot be reached in this case.

The case of the sphere function: Fast convergence ratio with $\lambda = 2d$. We point out here that for the specific case of the sphere function, a convergence ratio $\mathrm{CR}_\varepsilon = \Theta(1)$ can be reached with $\lambda = 2d$ in the domain $[0,1]^d$ by some algorithm of type full ranking (μ, λ)-ES.

This convergence ratio is easily obtained with the following algorithm. Let e_i denote the vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ with a unique 1 in position i. First split $[0,1]^d$ into the 2^d cells delimited by the d hyperplanes of equations $x_i = 1/2$; the full ranking of the $2d$ points $\{(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}) + \frac{\eta}{2}e_i \mid 1 \leqslant i \leqslant n,\ \eta \in \{-1, 1\}\}$ allows to decide in which of these cells the optimum lies; then the algorithm proceeds recursively. This is quite similar to the Hooke and Jeeves algorithm [12].

After n iterations, the point $x_n^{(f)}$ proposed by this algorithm satisfies $\|x_n^{(f)} - f^*\|_2 \leqslant \sqrt{d}/2^n$. Moreover, this distance is realized by some fitness functions. It follows that $n_{\varepsilon,\frac{1}{2}} = \log\frac{1}{\varepsilon} + \frac{1}{2}\log d$. On the other hand $\log(N(\varepsilon)) = \Theta(d\log\frac{1}{\varepsilon})$. Thus, we have obtained:

$$\text{For } \lambda = 2d:\ \mathrm{CR}_\varepsilon = \frac{\log N(\varepsilon)}{d\, n_{\varepsilon,\frac{1}{2}}} = \Theta(1). \tag{3}$$

4.3 Elitist Strategies

Results obtained in the case of (μ, λ) algorithms can be translated into the elitist setting. Bounds obtained in these cases are given in Figure 1 (Section 5). Proofs of these results are omitted due to space limitation.

5 Summary of Results

Let's apply the results obtained in the previous section to the simple framework of the domain $D = [0,1]^d$ with the Euclidean norm. Lower bounds obtained in this setting are summarized in Figure 1. Higher values mean better possible convergence ratios. However, it is not known when these convergence ratios can be achieved. Indeed, result marked with (*) in Figure 1 is improved in the special case of the sphere function in Section 4.2: this shows that at least in this case, general bounds on convergence ratio derived from VC-dimension are not tight. Discussion of these results follows.

	SB-(μ,λ)-ES	SB-$(\mu+\lambda)$-ES	Full ranking (μ,λ)-ES	Full ranking $(\mu+\lambda)$-ES	Full ranking $(\infty+\lambda)$-ES
CR	$\frac{V}{d}\log\lambda$	$\frac{V}{d}\log(\mu+\lambda)$	$\frac{V}{d}(\log(\lambda)+4\mu)$ (*)	$\frac{V}{d}(\log(\lambda+\mu)+4\mu)$	$\frac{4V\lambda}{d}$

Fig. 1. Upper bound on the convergence ratio in the case of Euclidean norm in the domain $[0,1]^d$, when the level sets of fitness functions have VC-dimension V

Asymptotic speed-up in the case of selection only, non-elitist. In the case of evolution strategies based on selection only (algorithms of type SB-(μ,λ)-ES), the linear speed-up of selection-based evolution strategies shown in [5] cannot be obtained for λ large enough. Asymptotically, the speed-up obtained with such an algorithm is at most logarithmic as shown in Theorem 2.

Selection based algorithms vs. full ranking. When moving from selection based algorithms of type SB-(μ,λ)-ES to full ranking (μ,λ)-ES, upper bounds on the convergence ratio obtained here in the general case do not forbid a strong improvement asymptotically; essentially, the speed-up that could be achieved moves from logarithmic to linear in λ.

However, we know from Proposition 1 that the speed-up is at most logarithmic for a full ranking (μ,λ)-ES in the special case of sphere function. This raises the following question: for which kind of fitness functions is it interesting to keep the full ranking information?

Acknowledgements. Many thanks to Anne Auger, Nikolaus Hansen and Fabien Teytaud for constructive talks. This work was partially supported by the Pascal Network of Excellence.

References

1. Arnold, D.V.: Optimal weighted recombination. In: Wright, A.H., Vose, M.D., De Jong, K.A., Schmitt, L.M. (eds.) FOGA 2005. LNCS, vol. 3469, pp. 215–237. Springer, Heidelberg (2005)
2. Auger, A.: Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible Markov chains. Theoretical Computer Science 334(1-3), 35–69 (2005)

3. Bäck, T., Hoffmeister, F., Schwefel, H.-P.: Extended selection mechanisms in genetic algorithms. In: Belew, R.K., Booker, L.B. (eds.) Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 92–99. Morgan Kaufmann Publishers, San Mateo (1991)
4. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, pp. 14–21. Lawrence Erlbaum Associates, Inc., Mahwah (1987)
5. Beyer, H.-G.: Toward a theory of evolution strategies: On the benefit of sex - the $(\mu/\mu, \lambda)$-theory. Evolutionary Computation 3(1), 81–111 (1995)
6. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: a comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
7. Devroye, L., Györfi, L., Lugosi, G.: A probabilistic Theory of Pattern Recognition. Springer, Heidelberg (1997)
8. Droste, S.: Not all linear functions are equally difficult for the compact genetic algorithm. In: Proc. of the Genetic and Evolutionary Computation COnference (GECCO 2005), pp. 679–686 (2005)
9. Feller, W.: An introduction to Probability Theory and its Applications. Wiley, Chichester (1968)
10. Gelly, S., Ruette, S., Teytaud, O.: Comparison-based algorithms are robust and randomized algorithms are anytime. Evolutionary Computation Journal (MIT Press), Special issue on bridging Theory and Practice 15(4), 411–434 (2007)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
12. Hooke, R., Jeeves, T.A.: "Direct search" solution of numerical and statistical problems. Journal of the ACM 8(2), 212–229 (1961)
13. Jägersküpper, J., Witt, C.: Rigorous runtime analysis of a $(\mu+1)$ES for the sphere function. In: GECCO, pp. 849–856 (2005)
14. Matoušek, J.: Lectures on Discrete Geometry. Graduate Texts in Mathematics, vol. 212. Springer, Heidelberg (2002)
15. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
16. Rudolph, G.: Convergence rates of evolutionary algorithms for a class of convex objective functions. Control and Cybernetics 26(3), 375–390 (1997)
17. Sauer, N.: On the density of families of sets. Journal of Combinatorial Theory, Ser. A 13(1), 145–147 (1972)
18. Teytaud, O., Gelly, S.: General lower bounds for evolutionary algorithms. In: Proceedings of PPSN, pp. 21–31 (2006)
19. Vapnik, V.N., Chervonenkis, A.Ya.: On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications XVI(2), 264–280 (1971)
20. Whitley, D.: The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In: Schaffer, J.D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 116–121. Morgan Kaufmann, San Mateo (1989)

Rigorous Runtime Analysis of Inversely Fitness Proportional Mutation Rates

Christine Zarges

Fakultät für Informatik, TU Dortmund, 44221 Dortmund, Germany
christine.zarges@tu-dortmund.de

Abstract. Artificial Immune Systems (AIS) are an emerging new field of research in Computational Intelligence that are applied to many areas of application, e.g., optimization, anomaly detection and classification. For optimization tasks, the use of hypermutation operators constitutes a common concept in AIS. By now, only little theoretical work has been done in this field. In this paper, we present a detailed theoretical runtime analysis that gives an insight into the dynamics of fitness based hypermutation processes. Two specific mutation rates are considered using a simple immune inspired algorithm. Our main focus lies thereby on the influence of parameters embedded in popular immune inspired hypermutation operators from the literature. Our theoretical findings are accompanied by some empirical results.

1 Introduction

Immune Algorithms (IAs) are a special class of biologically inspired algorithms, which are based on the immune system of vertebrates. In contrast to Evolutionary Algorithms (EAs), which emerged from a single main concept, IAs derive from various immunological theories, namely the clonal selection principle, negative selection, immune networks or the danger theory [1,2,3].

The field of IAs is a relatively new area of research, but has achieved various promising results in different areas of application. Like EAs, they are often applied to optimization. In this paper, we focus on algorithms based on the clonal selection principle [4], a theory used to describe the basic features of an adaptive immune response to invading pathogens (antigens). Due to this theory only immune cells, which recognize an antigen, proliferate and undergo a hypermutation process called affinity maturation.

During the last years, many clonal selection algorithms to tackle optimization problems have been developed, like, e.g., CLONALG [5], OPT-IA [6], the B-Cell-Algorithm [7] and MISA [8]. All these algorithms are population based. The input is usually represented by a population of antigens; a population of immune cells represents candidate solutions of the considered problem. During the clonal selection process, the clone rate of an immune cell is proportional to its fitness, i.e., its affinity to the presented antigen, whereas the mutation rate is inversely proportional to the fitness.

Until now, little theoretical work has been done for IAs. As pointed out by Timmis [9] and Hart and Timmis [10] one challenge for the future of IAs is the development of a theoretical basis for IAs as much work so far has been concentrated on direct application of known immune principles. Timmis et al. give an overview on existing theoretical work in the field of IAs [11]. For clonal selection algorithms the work done mainly covers convergence analysis [12,13,14]. Thus, the analysis of the performance of IAs on specific problems and the impact of mutation operators or other features of the algorithms is stated to be a "major theoretical challenge for the future" [11]. This work contributes a theoretical analysis for hypermutation operators frequently employed in IAs.

In the field of EAs often a constant, standard mutation rate of $1/n$, where n is the length of the bit string, is used. Jansen and Wegener [15] showed that using an appropriate mutation rate is essential for the performance of the algorithms. Jansen and Sudholt [16] and Doerr et al. [17] investigated the use of asymmetric mutation operators. In this paper, we theoretically analyze the runtime of inversely fitness proportional hypermutation operators on a simple toy problem. Note, that these operators are somehow similar to evolution strategies [18] as they employ self-adapting mechanisms.

The paper is organized as follows. In Section 2 we describe a simple IA and the framework for our analysis. Section 3 introduces a hamming distance based mutation rate while in Section 4 we present the theoretical analysis for immune inspired hypermutation operators. We complement our analysis by some experimental results in Section 5. Conclusion and proposals for future work are made in Section 6.

2 A Simple Immune Algorithm

In this section, we describe a simple IA based on the clonal selection theory that maximizes some objective function $f : \{0,1\}^n \to \mathbb{R}$. As we focus on the effect of mutation rates that are inversely proportional to the fitness of the current individual, we omit other possible operators (e.g., cloning or aging), reducing the size of our population of immune cells to one. This leads to the following simple IA that due to its similarity to the (1+1) EA [19] we call the (1+1) IA:

Algorithm 1. ((1+1) Immune Algorithm)

```
1. Choose $x \in \{0,1\}^n$ uniformly at random.
2. Create offspring $y := x$.
3. Flip each bit in $y$ independently with probability $\alpha(f(x))$.
4. If $f(y) \geq f(x)$: Set $x := y$.
5. Continue at 2.
```

We analyze how long the (1+1) IA takes to find an optimal solution starting from a randomly initialized solution. For this purpose we have chosen a classical toy problem for evolutionary algorithms, in which the objective is to maximize the number of ones in a bit string:

$$\text{ONEMAX}(x) = \sum_{i=1}^{n} x_i, \tag{1}$$

where n is the length of the bit string x. The (1+1) EA with mutation probability $1/n$ solves ONEMAX in time $\Theta(n \log n)$ [19].

3 A Hamming Distance Based Mutation Rate

We start our analysis with a simple, hamming distance based mutation rate $\alpha_h(v)$, which is defined as

$$\alpha_h(v) = \frac{n-v}{n}, \tag{2}$$

where n is the length of the bit string x. Let $v = f(x)$. Then, $n - v$ equals the hamming distance between the fitness of an individual x and the optimal fitness function value of ONEMAX. As the expected number of flipping bits for this operator is $n - v$, such a mutation rate is optimal if the objective is to maximize the probability to find the global optimum of ONEMAX in a single mutation step.

We first prove that with a probability close to 1, the (1+1) IA with hamming distance based mutation rate will not find the optimum of ONEMAX within a polynomial number of iterations. Let M_0 be the number of flipping zeros and M_1 the number of flipping ones, respectively. Then, for $k = n - v$ we have $E(M_0) = \frac{k^2}{n}$ and $E(M_1) = k - \frac{k^2}{n}$. Thus, the expected progress equals $E(M_0 - M_1) = \frac{2 \cdot k^2}{n} - k$. Note, that for an individual x with $f(x) \le \frac{n}{2}$ we have $\alpha_h(f(x)) \ge \frac{1}{2}$.

Theorem 1. *The probability that the (1+1) IA with hamming distance based mutation rate $\alpha_h(v) = \frac{n-v}{n}$ will not find the optimum of ONEMAX within $e^{c \cdot n}$ iterations ($c > 0$ constant, sufficiently small) is bounded below by $1 - e^{-\Omega(n)}$.*

Proof. By Chernoff bounds [20], we have $\text{Prob}(\frac{n}{3} \le f(x_0) \le \frac{2n}{3}) = 1 - e^{-\Omega(n)}$ for the initial bit string x_0. Then, as long as $f(x) \le \frac{2n}{3}$, we have $\text{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n)}$ as the following case inspection shows.

Case 1. For $(\frac{1}{2} + \delta) \cdot n \le f(x) \le \frac{2n}{3}$ ($0 < \delta < \frac{1}{6}$ constant) the probability to have $f(y) > f(x)$ is bounded above by $\text{Prob}(M_0 > M_1) \le \text{Prob}(M_0 > (1 + \delta) \cdot E(M_0)) \cdot \text{Prob}(M_1 < (1 - \delta) \cdot E(M_1)) = e^{-\Omega(n)}$ as for $k = n - f(x) \le (\frac{1}{2} - \delta) \cdot n$

$$(1+\delta) \cdot E(M_0) - (1-\delta) \cdot E(M_1) = (1+\delta) \cdot \tfrac{k^2}{n} - (1-\delta) \cdot \left(k - \tfrac{k^2}{n}\right)$$
$$= k \cdot \left(\frac{2k}{n} + \delta - 1\right) \le k \cdot \left(\frac{2 \cdot (\frac{1}{2} - \delta) \cdot n}{n} + \delta - 1\right) = -\delta \cdot k < 0$$

Case 2. For $\frac{n}{3} \le f(x) \le (\frac{1}{2} - \delta) \cdot n$ ($0 < \delta < \frac{1}{6}$ constant) the probability for having $f(y) > \frac{2n}{3}$ is maximal for $f(x) = \frac{n}{3}$. For this case, however, again Chernoff bounds yield $\text{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n)}$ as $E(M_0) = \frac{n}{9}$ and $E(M_1) = \frac{2n}{9}$.

Thus, we have $\text{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n)}$ if $\frac{n}{3} \le f(x_0) \le \frac{2n}{3}$. Together this shows that with probability $1 - e^{-\Omega(n)}$ even in $e^{c \cdot n}$ iterations the global optimum of ONEMAX is not found ($c > 0$ constant, sufficiently small). □

We have shown that the hamming distance based mutation rate $\alpha_h(v) = \frac{n-v}{n}$ with a randomly chosen initial bit string x_0 yields an exponential lower bound for the optimization time and thus, we get an exponential lower bound for the (1+1) IA with this operator. Suppose the initial bit string is located in some specific region of the search space. The next theorem proves that then, we can achieve a polynomial time bound with probability converging to 1 for $n \to \infty$.

Theorem 2. *If $f(x_0) = O(\log n)$ or $f(x_0) = n - O(\log n)$ for the initial bit string x_0, the (1+1) IA with hamming distance based mutation rate $\alpha_h(v) = \frac{n-v}{n}$ will find the optimum of ONEMAX within a polynomial number of iterations with probability $1 - n^{-\omega(1)}$.*

Proof. Let $M = M_0 + M_1$ and $\overline{M} = n - M$. For $f(x_0) = O(\log n)$ we have $E(\overline{M}) = O(\log n)$. Suppose $M_1 = O(\log n)$ and $\overline{M} = \omega(\log n)$. By Chernoff, the probability that $(1 + (\omega(1) - 1)) \cdot O(\log n) = \omega(\log n)$ bits do not flip is bounded above by $n^{-\omega(1)}$. Thus, we have $f(y) = n - O(\log n)$ with probability $1 - n^{-\omega(1)}$.

For $f(x_0) = n - O(\log n)$ we estimate the success probability, i.e., the probability to flip more zeros than ones, as follows. Let $k = n - f(x_0)$. Then,

$$\begin{aligned}
&P(M_0 > M_1)\\
&= \sum_{i=1}^{k} \binom{k}{i} \cdot \left(\frac{k}{n}\right)^{i} \cdot \left(1 - \frac{k}{n}\right)^{k-i} \cdot \left(\sum_{j=0}^{i-1} \binom{n-k}{j} \cdot \left(\frac{k}{n}\right)^{j} \cdot \left(1 - \frac{k}{n}\right)^{n-k-j} \right)\\
&\geq \sum_{i=1}^{k} \binom{k}{i} \cdot \left(\frac{k}{n}\right)^{i} \cdot \left(1 - \frac{k}{n}\right)^{k-i} \cdot \left(\sum_{j=0}^{i-1} \left(\frac{n-k}{k}\right)^{j} \cdot \left(\frac{k}{n}\right)^{j} \cdot \left(1 - \frac{k}{n}\right)^{n-k-j} \right)\\
&= \sum_{i=1}^{k} \binom{k}{i} \cdot \left(\frac{k}{n}\right)^{i} \cdot \left(1 - \frac{k}{n}\right)^{k-i} \cdot \left(\sum_{j=0}^{i-1} \left(1 - \frac{k}{n}\right)^{n-k} \right)\\
&= \left(1 - \frac{k}{n}\right)^{n-k} \cdot \sum_{i=1}^{k} i \cdot \binom{k}{i} \cdot \left(\frac{k}{n}\right)^{i} \cdot \left(1 - \frac{k}{n}\right)^{k-i} \geq \left(\frac{1}{e}\right)^{k} \cdot \frac{k^2}{n}
\end{aligned}$$

Thus, with $k = O(\log n) \Leftrightarrow f(x_0) = n - O(\log n)$ or $f(x_0) = O(\log n)$ we find the global optimum in $O(n^{O(1)})$ iterations with probability $1 - n^{-\omega(1)}$. □

4 Analysis of Immune Inspired Mutation Rates

Different types of immune inspired hypermutation operators have been proposed in the literature, e.g., proportional hypermutation, inversely proportional hypermutation and Hypermacromutation [6]. As already mentioned, in this paper, we focus on mutations rates, which are inversely proportional to the fitness of the current individual. Let

$$\alpha_1(v) = e^{-\rho \cdot v} \qquad \text{and} \qquad \alpha_2(v) = \tfrac{1}{\rho} \cdot e^{-v}, \tag{3}$$

where ρ is a parameter that controls the decay of the inverse exponential function and v is a fitness function value normalized in $[0, 1]$. The first equation has been originally introduced for CLONALG [5]. The second one has been proposed for an optimization version of aiNet [21]. Both functions have been analyzed experimentally on ONEMAX for a bit string length of $n = 100$ [22]. In this paper, we focus on the theoretical analysis of $\alpha_1(v)$ as a better performance for this mutation type on ONEMAX has been reported [22].

In population based IAs the fitness function value is normalized by dividing the fitness of an individual by the fitness of the best current individual (for a maximization problem). As our population consists of a single individual, we use the optimal value n of the considered objective function and thus obtain:

$$\alpha_1'(v) = e^{-\rho \cdot \frac{v}{n}} \tag{4}$$

In real applications one could alternatively use an upper bound for the optimal value. Note, that using an upper bound for the optimum leads to larger mutation rates, whereas the use of the fitness of the best current individual in a population yields smaller mutation rates.

The following theorems show that the choice of the parameter ρ is essential for the performance of the mutation operator. We first analyze the two extreme cases $\rho = 1$ and $\rho = \Omega(n)$ and prove that with a probability close to 1 for both parameters the (1+1) IA with mutation rate $\alpha_1'(v) = e^{-\rho \cdot \frac{v}{n}}$ will not find the optimal value of ONEMAX within a polynomial number of iterations.

Let again M_0 be the number of flipping zeros and M_1 the number of flipping ones, respectively. Then, for an individual x we have $E(M_0) = (n - f(x)) \cdot e^{-\rho \cdot \frac{f(x)}{n}}$ and $E(M_1) = f(x) \cdot e^{-\rho \cdot \frac{f(x)}{n}}$. The expected progress equals $(n - 2 \cdot f(x)) \cdot e^{-\rho \cdot \frac{f(x)}{n}}$.

Theorem 3. *The probability that the (1+1) IA with mutation rate $\alpha_1'(v) = e^{-\rho \cdot \frac{v}{n}}$ and $\rho = 1$ will not find the optimum of ONEMAX within $e^{c \cdot n}$ iterations ($c > 0$ constant, sufficiently small) is bounded below by $1 - e^{-\Omega(n)}$.*

Proof. The proof follows the line of thought of Theorem 1. Again, we have $\mathrm{Prob}(\frac{n}{3} \le f(x_0) \le \frac{2n}{3}) = 1 - e^{-\Omega(n)}$ for the initial bit string x_0 due to Chernoff bounds. Then, analogous to Theorem 1, we show $\mathrm{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n)}$ as long as $f(x) \le \frac{2n}{3}$.

Case 1. For $(\frac{1}{2} + \delta) \cdot n \le f(x) \le \frac{2n}{3}$ ($0 < \delta < \frac{1}{6}$ constant) we have

$$\begin{aligned}&(1+\delta)E(M_0) - (1-\delta)E(M_1) = e^{-\frac{f(x)}{n}} \cdot [(1+\delta) \cdot (n - f(x)) - (1-\delta) \cdot f(x)] \\ &\le e^{-\frac{f(x)}{n}} \cdot \left[(1+\delta) \cdot (\tfrac{1}{2} - \delta) \cdot n - (1-\delta) \cdot (\tfrac{1}{2} + \delta) \cdot n\right] \le -\delta \cdot n < 0\end{aligned}$$

and thus, the probability to have $f(y) > f(x)$ is bounded above by $\mathrm{Prob}(M_0 > M_1) \le \mathrm{Prob}(M_0 > (1+\delta) \cdot E(M_0)) \cdot \mathrm{Prob}(M_1 < (1-\delta) \cdot E(M_1)) = e^{-\Omega(n)}$.

Case 2. For $\frac{n}{3} \le f(x) \le (\frac{1}{2} - \delta) \cdot n$ ($0 < \delta < \frac{1}{6}$ constant) again the probability for having $f(y) > \frac{2n}{3}$ is maximal for $f(x) = \frac{n}{3}$. In this case, Chernoff bounds yield $\mathrm{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n)}$ as $E(M_0) = \frac{2n}{3 \cdot \sqrt[3]{e}}$ and $E(M_1) = \frac{n}{3 \cdot \sqrt[3]{e}}$.

Thus, we have $\text{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n)}$ if $\frac{n}{3} \leq f(x_0) \leq \frac{2n}{3}$ and the theorem follows. $\square$

We remark, that for $\rho = 1$ we cannot show a polynomial bound for specific regions of the search space (cf. Theorem 2), as, e.g., for $f(x_0) = n - O(\log n)$ or even $f(x_0) = n - O(1)$ the expected progress equals $O(\log n) - n$ and $O(1) - n$, respectively. The results above hold for all $\rho = O(1)$. Thus, we can conclude that the parameter ρ should depend on the length n of the bit string.

Theorem 4. *For $n \to \infty$ the (1+1) IA with mutation rate $\alpha_1'(v) = e^{-\rho \cdot \frac{v}{n}}$ and $\rho = \Omega(n)$ yields an exponential expected waiting time for a mutation actually changing a bit.*

Proof. With $\rho = \Omega(n)$ the mutation rate equals $\alpha_1'(v) = e^{-\Omega(n) \cdot \frac{v}{n}} = e^{-\Omega(v)}$. By Chernoff bounds, we have $\text{Prob}(\frac{n}{3} \leq f(x_0) \leq \frac{2n}{3}) = 1 - e^{-\Omega(n)}$ for the initial bit string x_0. The mutation rate then equals $\alpha_1'(v) = e^{-\Omega(n)}$. Thus, the expected number of flipping bits in an iteration is $\frac{n}{e^{\Omega(n)}}$ and the theorem follows. $\square$

Note, that already for $f(x_0) = \omega(\log n)$ we have a mutation rate $\alpha_1' = e^{-\omega(\log n)} = n^{-\omega(1)}$. Then, the expected number of flipping bits in an iteration is $\frac{n}{n^{\omega(1)}}$, converging to 0 for $n \to \infty$.

We have seen, that on the one hand for $\rho = \Omega(n)$ the mutation rate $\alpha_1'(v) = e^{-\rho \cdot \frac{v}{n}}$ becomes exponentially small and yields an exponential expected waiting time for a mutation actually changing a bit. On the other hand for $\rho = 1$ we will not find the optimum of ONEMAX within a polynomial number of iterations with probability converging to 1 exponentially fast. The latter results from the fact that for $f(x) > \frac{n}{2}$ and $f(x) \to n$ we have a large negative drift and additionally for the relevant values of $f(x)$ the probability for a one bit mutation is exponentially small as $\frac{1}{e} \leq \alpha_1'(v) \leq 1$.

For this reason, we now choose a value for ρ, which is in between these two extreme cases, namely $\rho = \ln n$. For $\rho = \ln n$, we have $\alpha_1'(v) = n^{-\frac{v}{n}}$ and thus, $\frac{1}{n} \leq \alpha_1'(v) < \frac{1}{2}$ if $v > \frac{n}{\log}$. Hence, in this case we get, in contrast to $\rho = 1$ and $\rho = \Omega(n)$, reasonable values for the mutation probability.

The next theorem shows that for $\rho = \ln n$ we get indeed a similar result as for $\rho = 1$, but in praxis $\rho = \ln n$ shows a much better performance (cf. Section 5) as the probability that the (1+1) IA will not find the optimum within a polynomial number of iterations converges much more slowly to 1 than it does for $\rho = 1$.

Theorem 5. *The probability that the (1+1) IA with mutation rate $\alpha_1'(v) = n^{-\frac{v}{n}}$ will not find the optimum of ONEMAX within $e^{d \cdot n^c}$ iterations is bounded below by $1 - e^{-\Omega(n^c)}$ (for constants $0 < c < \frac{1}{2}$ and $d > 0$ sufficiently small).*

Proof. The proof follows the line of thought of Theorem 1. Again, we have $\text{Prob}(\frac{n}{3} \leq f(x_0) \leq \frac{2n}{3}) = 1 - e^{-\Omega(n)}$ for the initial bit string x_0 due to Chernoff bounds. Let $f(x) = c \cdot n$. Then, $E(M_0) = (1 - c) \cdot n^{1-c}$ and $E(M_1) = c \cdot n^{1-c}$.

Analogous to Theorem 1 we show $\text{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n^c)}$ ($0 < c < \frac{1}{2}$ constant) as long as $f(x) \leq \frac{2n}{3}$ by the following case inspection.

Case 1. Similarly to Theorem 3 we can prove that

$$(1+\delta) \cdot E(M_0) - (1-\delta) \cdot E(M_1) \leq -\frac{\delta \cdot n}{n^{\frac{f(x)}{n}}} \leq -\frac{\delta \cdot n}{n^{\frac{1}{2}+\delta}} = -\delta \cdot n^{\frac{1}{2}-\delta} < 0$$

and thus, for $(\frac{1}{2} + \delta) \cdot n \leq f(x) \leq \frac{2n}{3}$ ($0 < \delta < \frac{1}{6}$ constant) the probability to have $f(y) > f(x)$ is bounded above by $\text{Prob}(M_0 > M_1) \leq \text{Prob}(M_0 > (1+\delta) \cdot E(M_0)) \cdot \text{Prob}(M_1 < (1-\delta) \cdot E(M_1)) = e^{-\Omega(n^{\frac{1}{2}-\delta})}$.

Case 2. For $\frac{n}{3} \leq f(x) \leq (\frac{1}{2} - \delta) \cdot n$ ($0 < \delta < \frac{1}{6}$ constant) the probability for having $f(y) > \frac{2n}{3}$ is again maximal for $f(x) = \frac{n}{3}$. For this case, Chernoff bounds yield $\text{Prob}(f(y) > \frac{2n}{3}) = e^{-\Omega(n)}$ as $E(M_0) = \frac{2n^{\frac{2}{3}}}{3}$, $E(M_1) = \frac{n^{\frac{2}{3}}}{3}$ and $\frac{n}{3} + (1+\varepsilon) \cdot E(M_0) - (1-\varepsilon) \cdot E(M_1) > \frac{2n}{3}$ for $\varepsilon > \frac{1}{3} \cdot (n^{\frac{1}{3}} - 1)$.

Together this shows that with probability $1 - e^{-\Omega(n^c)}$ even in $e^{d \cdot n^c}$ iterations the optimum of ONEMAX is not found ($0 < c < \frac{1}{2}$, $d > 0$ sufficiently small, constant). □

Similarly to our analysis in Section 3 we can show, that for some regions of the search space we achieve a polynomial time bound.

Theorem 6. *If $f(x_0) = n - O(\frac{n}{\log n})$ for the initial bit string x_0, the (1+1) IA with mutation rate $\alpha'_1(v) = n^{-\frac{v}{n}}$ will in expectation find the optimum of ONEMAX within a polynomial number of iterations.*

Proof. Let $k = n - f(x_0)$. For $k = O(\frac{n}{\log n})$ we get an upper bound for the mutation rate by $\frac{1}{n^{\frac{n-k}{n}}} = \frac{n^{\frac{k}{n}}}{n} = \frac{e^{\frac{\ln n \cdot k}{n}}}{n} = \frac{e^{O(1)}}{n}$. Moreover, we have the trivial lower bound $\frac{1}{n^{\frac{n-k}{n}}} \geq \frac{1}{n}$. The probability for a one bit mutation can then be estimated by

$$\binom{k}{1} \cdot \frac{1}{n^{\frac{n-k}{n}}} \cdot \left(1 - \frac{1}{n^{\frac{n-k}{n}}}\right)^{n-1} \geq \frac{k}{n} \cdot \left(1 - \frac{e^{O(1)}}{n}\right)^{n-1}$$
$$\geq \frac{k}{n} \cdot e^{-\frac{e^{O(1)} \cdot (n-1)}{n - e^{O(1)}}} \geq \frac{k}{n} \cdot e^{-e^{O(1)}} \overset{k=O(\frac{n}{\log n})}{=} \frac{O(1)}{e^{e^{O(1)}} \cdot \log n}$$

Thus, for $f(x_0) = n - O(\frac{n}{\log n})$ the optimum is found in expected $O(n)$ iterations and the theorem follows. □

From the above theorems we can conclude that on the one hand $\rho = \ln n$ yields a smaller negative drift and thus, the probability that the global optimum of ONEMAX is not found converges much more slowly to 1 than for $\rho = 1$ or the hamming distance based mutation rate $\alpha_h(v)$. Moreover, the regions of the search space, in which we have an expected polynomial optimization time, is larger for $\rho = \ln n$. Figure 1 visualizes this effect for different values of n, where $O(\log(n))$

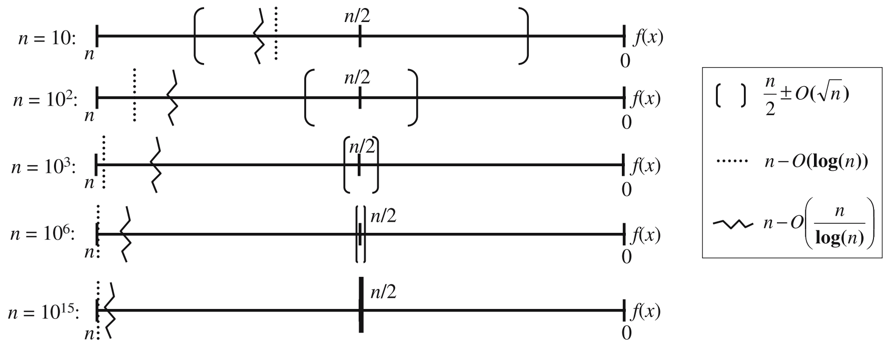


Fig. 1. The relative size of significant regions in the search space

corresponds to the hamming distance based mutation rate $\alpha_h(v)$ and $O(\frac{n}{\log n})$ belongs to $\alpha'_1(v)$ with $\rho = \ln n$. By Chernoff bounds, the probability that the initial bit string x_0 contains $\frac{n}{2} \pm O(\sqrt{n})$ ones is bounded below by $1 - e^{-\omega(1)}$. Numerical values for functions $O(f(n))$ are obtained by simply calculating $f(n)$.

We recognize that for small values of n, we have $\frac{n}{2} - \sqrt{n} < \frac{n}{\log n} < \log n$, whereas for large n we have the reverse order as $O(\log n) = O(\frac{n}{\log n}) = \frac{n}{2} - O(\sqrt{n})$. As our results are asymptotical, the behavior presented in the foregoing theorems rather corresponds to large n and thus, for n small enough the (1+1) IA yields suitable performance that we further investigate in the next section.

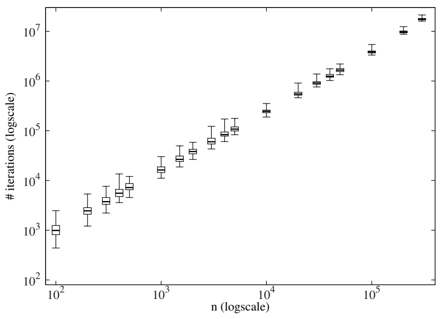
We remark, that for $\alpha_2(v) = \frac{1}{\rho} \cdot e^{-v}$, where v is a fitness function value normalized in $[0, 1]$ by the approach described above, probably similar results can be shown. For $\rho = 1$ we have $\alpha_1(v) = \alpha_2(v)$ while $\rho = \Omega(n)$ yields mutation probabilities $< \frac{1}{n}$ for $k \le n - \ln(1/c) \cdot n$ ($0 < c < 1$ const.) and finally for $\rho = \ln n$ we have $\frac{1}{e \cdot \ln n} \le \alpha_2(v) \le \frac{1}{\ln n}$. Due to this properties of the mutation probability we expect a worse behavior in accordance to the results previously reported [22].

5 Experimental Results

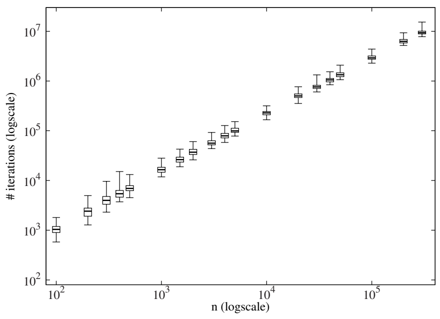
Finally, we present some empirical results for the mutation operators discussed in the foregoing sections. In our experiments we studied how many iterations are actually needed to optimize ONEMAX for a given bit string length n. For each n 100 independent runs were performed.

Figure 2(a) visualizes the median, lower and upper quartile as well as the smallest and largest value observed for the (1+1) IA with mutation rate $\alpha'_1(v) = e^{-\rho \cdot \frac{v}{n}}$ and $\rho = \ln n$ using a boxplot diagram. For the purpose of comparison, we additionally illustrate the corresponding values for the (1+1) EA with standard mutation rate $1/n$ in Figure 2(b). Note, that in both plots we use logarithmic scales for x- and y-axis. We observe, that for $n \le 10^5$ the runtime of the (1+1) EA is quite similar to that of the (1+1) IA with $\alpha'_1(v) = e^{-\rho \cdot \frac{v}{n}}$ and $\rho = \ln n$, while afterwards the differences in runtime become obvious.

We omit a graphical presentation for the other mutation operators as for $\rho = 1$ already $n = 40$ leads to approximately 10^8 iterations. For $\rho = n$ the performance



(a) Mutation rate $\alpha'_1(v) = n^{-\frac{v}{n}}$



(b) Standard mutation $1/n$

Fig. 2. Experimental results for different bit string lengths n

is even worse as around 10^8 iterations are needed for $n = 20$. With $\rho = n/2$ we have again 10^8 iterations if $n = 40$. For the hamming distance based mutation rate $\alpha_h(v) = \frac{n-v}{n}$ a bit string length of $n = 300$ results in 10^9 iterations. Note, that with $\alpha'_1(v) = e^{-\rho\cdot\frac{v}{n}}$ and $\rho = \ln n$ or the (1+1) EA with standard mutation rate $1/n$, we can solve instances of $n = 10^5$ in approximately 10^6 iterations. Thus, from a practical point of view, the (1+1) IA yields comparable performance for $\alpha'_1(v)$ and $\rho = \ln n$ if n is not too large, while the other hypermutation operators are unsuitable for the problem considered in this paper.

6 Conclusion and Future Work

In this paper, we presented a detailed theoretical runtime analysis for different inversely fitness proportional mutation operators. We introduced a simple immune inspired algorithm and proved that maximizing the probability to find the global optimum in a single iteration yields an exponential optimization time. Furthermore, we showed that for immune inspired mutation rates the choice of the parameter ρ is essential as an inappropriate value leads to unreasonable mutation probabilities. Although we proved asymptotically exponential runtimes for all mutation rates considered, our experimental analysis shows that with an appropriate parameter choice the performance of the (1+1) IA is comparable to that of the (1+1) EA with standard mutation rate $1/n$ if n is not too large.

Our results contribute a first insight into the behavior of inversely fitness proportional hypermutation operators. Nevertheless, much work remains for the future. We did not consider Hypermacromutation and mutation potentials [6]. Moreover, an analysis for population based algorithms and other toy or real world problems is necessary to get a deeper understanding of the underlying dynamics. In particular, dynamic problems seem to be worthwhile as IAs showed promising results in these applications.

Acknowledgment

I thank Thomas Jansen for his valuable advice and help while preparing this paper.

References

1. Dasgupta, D. (ed.): Artificial Immune Systems and Their Applications. Springer, Heidelberg (1998)
2. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, Heidelberg (2002)
3. Timmis, J., Andrews, P.S., Owens, N., Clark, E.: An interdisciplinary perspective on artificial immune systems. Evolutionary Intelligence 1, 5–26 (2008)
4. Burnet, F.M.: The Clonal Selection Theory of Acquired Immunity. Cambridge University Press, Cambridge (1959)
5. de Castro, L.N., Zuben, F.J.V.: Learning and Optimization Using the Clonal Selection Principle. IEEE Trans. on Evol. Comp. 6(3), 239–251 (2002)
6. Cutello, V., Nicosia, G., Pavone, M.: Exploring the Capability of Immune Algorithms: A Characterization of Hypermutation Operators. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 263–276. Springer, Heidelberg (2004)
7. Kelsey, J., Timmis, J.: Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 207–218. Springer, Heidelberg (2003)
8. Cortés, N.C., Coello, C.A.C.: Multiobjective Optimization Using Ideas from the Clonal Selection Principle. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 158–170. Springer, Heidelberg (2003)
9. Timmis, J.: Artificial Immune Systems – Today and Tomorrow. Natural Computing 6(1), 1–18 (2007)
10. Hart, E., Timmis, J.: Application Areas of AIS: The Past, the Present and the Future. Applied Soft Computing 8(1), 191–201 (2008)
11. Timmis, J., Hone, A., Stibor, T., Clark, E.: Theoretical Advances in Artificial Immune Systems. Theoretical Computer Science (in press, 2008)
12. Villalobos-Arias, M., Coello Coello, C.A., Hernández-Lerma, O.: Convergence analysis of a multiobjective artificial immune system algorithm. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 226–235. Springer, Heidelberg (2004)
13. Clark, E., Hone, A., Timmis, J.: A Markov Chain Model of the B-Cell Algorithm. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 318–330. Springer, Heidelberg (2005)
14. Cutello, V., Nicosia, G., Romeo, M., Oliveto, P.S.: On the Convergence of Immune Algorithms. In: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), pp. 409–415. IEEE Press, Los Alamitos (2007)
15. Jansen, T., Wegener, I.: On the Choice of the Mutation Probability for the (1+1) EA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 89–98. Springer, Heidelberg (2000)

16. Jansen, T., Sudholt, D.: Design and analysis of an asymmetric mutation operator. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005). IEEE Press, Los Alamitos (2005)
17. Doerr, B., Hebbinghaus, N., Neumann, F.: Speeding Up Evolutionary Algorithms through Asymmetric Mutation Operators. Evol. Comput. 15(4), 401–410 (2007)
18. Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley & Sons, Chichester (1995)
19. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276(1-2), 51–81 (2002)
20. Hagerup, T., Rüb, C.: A guided tour of Chernoff bounds. Information Processing Letters 33, 305–308 (1990)
21. de Castro, L.N., Timmis, J.: An Artificial Immune Network for Multimodal Function Optimization. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002), pp. 674–699. IEEE Press, Los Alamitos (2002)
22. Cutello, V., Narzisi, G., Nicosia, G., Pavone, M.: Clonal selection Algorithms: A Comparative Case Study Using Effective Mutation Potentials. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 13–28. Springer, Heidelberg (2005)

Covariance Matrix Adaptation Revisited – The CMSA Evolution Strategy –

Hans-Georg Beyer[1] and Bernhard Sendhoff[2]

[1] Vorarlberg University of Applied Sciences
Hochschulstr. 1, A-6850 Dornbirn, Austria
hans-georg.beyer@fh-vorarlberg.ac.at
[2] Honda Research Institute Europe GmbH
Carl-Legien-Str. 30, D-63073 Offenbach/Main, Germany
bs@honda-ri.de

Abstract. The covariance matrix adaptation evolution strategy (CMA-ES) rates among the most successful evolutionary algorithms for continuous parameter optimization. Nevertheless, it is plagued with some drawbacks like the complexity of the adaptation process and the reliance on a number of sophisticatedly constructed strategy parameter formulae for which no or little theoretical substantiation is available. Furthermore, the CMA-ES does not work well for large population sizes. In this paper, we propose an alternative – simpler – adaptation step of the covariance matrix which is closer to the "traditional" mutative self-adaptation. We compare the newly proposed algorithm, which we term the CMSA-ES, with the CMA-ES on a number of different test functions and are able to demonstrate its superiority in particular for large population sizes.

1 Introduction

State-of-the-art Evolutionary Algorithms (EA) in real-valued search domains use non-isotropic mutation distributions in order to explore the search space. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES), proposed by Hansen, Ostermeier, and Gawelczyk [1] and further developed in [2, 3], is currently the most widely used, and in its restart version [4] arguably the best performing EA for continuous optimization on a (sub-)set of test functions [5].[1]

At the same time, the CMA-ES is also plagued with a couple of drawbacks which we want to address in this paper by proposing an alternative adaptation scheme incorporating mutative self-adaptation. As we will see in the next section, the adaptation process in the CMA-ES is rather complex and involves a number of free parameters which have to be set with no or little theoretical guidance. Although thorough empirical investigations have been performed to identify suitable parameter settings [2, 3], still the application of the algorithm relies on *ad hoc* rules.

Secondly, the performance of the CMA-ES does not scale well with increasing population size. This problem has been alleviated by the introduction of the hybrid version

[1] According to "Tutorial: Covariance Matrix Adaptation (CMA) Evolution Strategy", presented by N. Hansen at PPSN Conference, Sep. 8, 2006, Reykjavik.

of the CMA-ES [3] with direct covariance matrix estimation, which will be our starting point in the next section and which will be used for comparison with our suggested algorithm.

Additionally, due to the cumulative step size adaptation the CMA-ES experiences problems when the fitness information is disturbed by heavy noise (noisy objective functions) [6, 7] and instabilities can occur when very large populations are needed [8].

Extensions of the CMA-ES and alternative approaches to covariance matrix adaptation have been proposed in the literature. Auger et al. [9] proposed an alternative method to calculate the covariance matrix by locally estimating the Hessian matrix (Taylor expansion), however, at the expense of a large computational overhead of $\mathcal{O}(N^6)$. A first multi-objective $(1+\lambda)$-CMA-ES has been described in [10] that uses the ”traditional” 1/5-rule for controlling the global step size.

In this paper, we will proceed in a different direction and revisit the mutative self-adaptation process in the context of covariance matrix adaptation. In the next section, we will briefly recall the CMA-ES and propose our new algorithm in Section 3. The empirical comparison between both algorithms will be described in Section 4 followed by the conclusion in the last section.

2 The $(\mu/\mu_W, \lambda)$-CMA-ES

In Figure 1 the basic $(\mu/\mu_W, \lambda)$-CMA-ES is presented. This is done at a level that assumes that the reader is already acquainted with the (hybrid) CMA-ES as described in [3].

The CMA-ES uses weighted recombination which is indicated by the subscript “W” in the strategy parentheses. The correlated mutations are generated in a two-step process where at first a vector $\mathbf{N}_l(\mathbf{0}, \mathbf{I})$ of i.i.d. standard normal random components is transformed by the matrix $\sqrt{\mathbf{C}}$ in step (L1). The resulting random vectors $\mathbf{z} = \sqrt{\mathbf{C}}\,\mathbf{N}(\mathbf{0}, \mathbf{I})$ are $\mathbf{N}(\mathbf{0}, \mathbf{C})$ distributed. The matrix $\sqrt{\mathbf{C}}$ may be interpreted as the “square root” of the covariance matrix $\mathbf{C}$. The standard way in CMA-ES [2, 3] to obtain $\sqrt{\mathbf{C}}$ is based on eigenvalue decomposition solving the eigenvalue problem. After producing the correlated Gaussian vector $\mathbf{s}$, it is scaled in length in (L2), thus, representing the mutation $\sigma\mathbf{s}$ which is finally added to the old parental state producing the offspring in (L2). The offspring’s fitness is evaluated in (L3). The new parental state is calculated in (L4) by recombination of the μ best offspring realized by weighted averaging. The adaptation of $\mathbf{C}$ is performed in (L6) using a cumulated $\mathbf{p}$ vector and the generational cross momentum matrix estimate $\langle \mathbf{s}\mathbf{s}^{\mathrm{T}} \rangle_{\mathrm{w}}$ weighted by the μ_{eff}^{-1} factor. (L6) performs an exponential smoothing (averaging) where the $\mathbf{C}$ “memory” decays with $(1-\tau_{\mathrm{c}}^{-1})^g$ (g - generation counter). The quantity τ_{c} can be interpreted as a decay time constant determining the number of generations g needed to “forget” the initial $\mathbf{C}$ matrix. It is quite clear that τ_{c} must be a function of the endogenous strategy parameters and the problem dimensionality N. In (L5) exponential smoothing is used to update the $\mathbf{p}$ vector with the direction $\langle \mathbf{s} \rangle_{\mathrm{w}}$ of the actually taken step from parent $\mathbf{y}$ at generation g to $g+1$ which has taken place in (L4). Therefore, $\mathbf{p}$ may be regarded as the average search step. The update of the covariance matrix $\mathbf{C}$ via the $\mathbf{p}$ vector is done in such a way that *selected* steps from the past on average are also preferred in future. This resembles the *momentum term*

```
$(\mu/\mu_W, \lambda)$-CMA-ES (one generation cycle)

For $l = 1$ To $\lambda$
    $\mathbf{s}_l \leftarrow \sqrt{\mathbf{C}}\,\mathbf{N}_l(\mathbf{0}, \mathbf{I})$    (L1)
    $\mathbf{y}_l \leftarrow \mathbf{y} + \sigma \mathbf{s}_l$    (L2)
    $f_l \leftarrow f(\mathbf{y}_l)$    (L3)
End
$\mathbf{y} \leftarrow \mathbf{y} + \sigma \langle \mathbf{s} \rangle_\mathrm{w}$    (L4)
$\mathbf{p} \leftarrow \left(1 - \frac{1}{\tau_\mathrm{p}}\right) \mathbf{p} + \sqrt{\frac{1}{\tau_\mathrm{p}}\left(2 - \frac{1}{\tau_\mathrm{p}}\right)} \sqrt{\mu_\mathrm{eff}}\, \langle \mathbf{s} \rangle_\mathrm{w}$    (L5)
$\mathbf{C} \leftarrow \left(1 - \frac{1}{\tau_\mathrm{c}}\right) \mathbf{C} + \frac{1}{\tau_\mathrm{c}} \left[\frac{1}{\mu_\mathrm{eff}} \mathbf{p}\mathbf{p}^T + \left(1 - \frac{1}{\mu_\mathrm{eff}}\right) \langle \mathbf{s}\mathbf{s}^T \rangle_\mathrm{w}\right]$    (L6)
$\mathbf{p}_\sigma \leftarrow \left(1 - \frac{1}{\tau_\sigma}\right) \mathbf{p}_\sigma + \sqrt{\frac{1}{\tau_\sigma}\left(2 - \frac{1}{\tau_\sigma}\right)} \sqrt{\mu_\mathrm{eff}}\, \langle \mathbf{N}(\mathbf{0}, \mathbf{I}) \rangle_\mathrm{w}$    (L7)
$\sigma \leftarrow \sigma \exp\left[\frac{\|\mathbf{p}_\sigma\| - \overline{\chi_N}}{d\, \overline{\chi_N}}\right]$    (L8)
```

Fig. 1. The algorithmic "essence" of the CMA-ES. Endowed with initialization an outer generation loop and an appropriated termination condition, an approximation of the optimizer is given by the final result of the parent $\mathbf{y}$. In general weighted recombination, denoted by "$\langle \cdot \rangle_\mathrm{w}$", is used. Note, the individuals' $\mathbf{N}$ vectors used in (L7) are from the selected individuals that have been generated in (L1). $\overline{\chi_N} = \mathrm{E}[\chi_N]$ is the expected value of the χ distribution with N degrees of freedom being the search space dimensionality. Initially, $\mathbf{C} = \sqrt{\mathbf{C}} = \mathbf{I}$, $\mathbf{p}_\sigma = \mathbf{0}$, and $\mathbf{p} = \mathbf{0}$. Basically, the following parameters have to be chosen d, τ_σ, τ_c, τ_p, and μ_eff.

approach in nonlinear programming. Therefore, the original form of the CMA-ES [2] can also be regarded as a *randomized momentum term* strategy.

3 The $(\mu/\mu_I, \lambda)$-CMA-σ-SA-ES

There are two main ingredients to build an efficient ES that works well on arbitrarily rotated ellipsoidal success domains:

1. A covariance matrix adaptation algorithm which is able to learn the shape of the success domain sufficiently exact and fast,
2. A routine that adapts a global step size σ

As we mentioned already in the introduction, the disadvantage of the different versions of the CMA-ES presented in the literature is the large number of exogenous strategy parameters needed. There are five main parameters (d, τ_σ, τ_c, τ_p, and μ_eff^{-1}) interacting with each other dynamically. While the effect of d and τ_σ has been analyzed on the sphere model [11], the interaction with the other time constants remains unclear. Furthermore, the CMA-ES does not always behave well in robust optimization scenarios [8, 12] when the number of offspring λ is significantly larger than the parameter space dimension.

3.1 The $(\mu/\mu_I, \lambda)$-CMA-σ-SA-ES Algorithm

In the following, the CMSA-ES will be proposed based on a radical simplification of the covariance learning rule and a revival of the well-known σ-self-adaptation (σSA) approach. Figure 2 shows the contents of the generation loop. As customary in

```
(μ/μ_I, λ)-CMA-σ-SA-ES (one generation cycle)

For $l = 1$ To $\lambda$
    $\sigma_l \leftarrow \langle\sigma\rangle e^{\tau\mathcal{N}_l(0,1)}$                  (R1)
    $\mathbf{s}_l \leftarrow \sqrt{\mathbf{C}}\,\mathbf{N}_l(\mathbf{0}, \mathbf{I})$          (R2)
    $\mathbf{z}_l \leftarrow \sigma_l \mathbf{s}_l$                         (R3)
    $\mathbf{y}_l \leftarrow \mathbf{y} + \mathbf{z}_l$                       (R4)
    $f_l \leftarrow f(\mathbf{y}_l)$                          (R5)
End
$\mathbf{y} \leftarrow \mathbf{y} + \langle\mathbf{z}\rangle$                        (R6)
$\mathbf{C} \leftarrow \left(1 - \frac{1}{\tau_c}\right)\mathbf{C} + \frac{1}{\tau_c}\langle\mathbf{s}\mathbf{s}^T\rangle$   (R7)
```

Fig. 2. Contents of the generation loop of the self-adaptive CMA-ES. Recombination, expressed by the "$\langle\cdot\rangle$" notation, is done (in the simplest case) by mean value calculation. The covariance matrix is initially chosen to be the identity matrix, i.e. $\mathbf{C} = \sqrt{\mathbf{C}} = \mathbf{I}$. For the choice of the strategy parameters τ and τ_c, see the text.

self-adaptation ES, each of the λ offspring individuals has its own mutation strength σ_l which is generated by the log-normal rule in line (R1). The generation of the object parameter $\mathbf{y}_l$ is done consecutively in line (R2 – R4). First, correlated random direction $\mathbf{s}_l$ is generated in (R2). This random direction is scaled in length by the individual's mutation strength σ_l in (R3) and finally added to the parental state $\mathbf{y}$ in line (R4) producing the offspring's object parameter vector $\mathbf{y}$. Its fitness f_l is evaluated in (R5).

In line (R6), recombination of the μ best offspring is performed. In the experiments done so far, $w_m = 1/\mu$ appeared as a reasonable choice, i.e., the angular bracket operation $\langle\cdot\rangle$ is simply an averaging over the μ best offspring individuals.

The covariance matrix adaptation takes place in (R7). Comparing with the rules used in the hybrid CMA-ES in lines (L5) and (L6), Fig. 1, one sees how simple this new rule is. Actually, it could be recovered from (L6) for $\mu_{\text{eff}} \to \infty$. As will be shown in the experimental Section 4, this CMA rule together with σ-self-adaptation yields comparable and even better results. As in the case of the object parameter recombination, recombining the generational cross momentum matrices $\mathbf{s}_m\mathbf{s}_m^{\mathrm{T}}$ is done with uniform weights (i.e., simple averaging over the contribution of the μ best individuals).

Due to the simplicity of the newly proposed self-adaptive CMSA-ES, there is a certain chance to put the choice of the (only) two endogenous strategy parameters, the learning rate τ and the covariance cumulation time constant τ_{c}, on a theoretically motivated basis.

3.2 Parameter Settings for the CMSA-ES

The Learning Parameter $\boldsymbol{\tau}$. This parameter basically influences the time needed to learn the global step size σ and its accuracy. Assuming a locally ellipsoidal fitness landscape and provided that the covariance is adapted correctly, the $\sigma\mathbf{N}_l(\mathbf{0},\mathbf{I})$ vectors in the CMSA-ES of Fig. 2 "experience" conditions similar to a spherical landscape. That is, under steady state conditions, one can use the τ which maximizes the steady state progress rate on the sphere model. As can be shown (due to space restrictions the derivation is beyond the scope of this paper) for sufficiently large μ, λ, and N this is the case for

$$\tau_{\mathrm{opt}} = \frac{1}{\sqrt{2N}}. \tag{1}$$

This value has been used in the simulations of the CMSA-ES presented below. Note, this choice is *not* the optimal one for the initial phase of covariance adaptation. If one wants to increase the speed by which the $\mathbf{C}$ matrix is adapted, smaller values (e.g. $\tau = \tau_{\mathrm{opt}}/2$) should be used. A strategy that provides a "second order" adaptation of τ could be envisioned, but has not been tested yet.

The $\boldsymbol{\tau_{\mathrm{c}}}$ Time Constant. The covariance learning rule (R7) contains the covariance learning time constant τ_{c}, the choice of which can be derived by information theoretical means. There are two aspects that must be considered: (1) the information dynamics of the covariance update; and (2) the minimum information needed to determine a covariance matrix. Again we must defer the derivation steps to an upcoming paper. The final result of the derivation is

$$\tau_{\mathrm{c}} = 1 + \frac{N(N+1)}{2\mu}. \tag{2}$$

This formula will be used in the simulations of the CMSA-ES in Section 4.

An Alternative Approach to $\sqrt{\mathbf{C}}$. Calculating $\sqrt{\mathbf{C}}$ via spectral decomposition requires the solution of the eigenvalue problem. While that approach provides additional information w.r.t. the sensitivity of the fitness landscape in the vicinity of the optimizer state, it is computationally demanding and not always required. Dropping the symmetry of the $\sqrt{\mathbf{C}}$ matrix, the Cholesky decomposition offers a much simpler alternative which does not need the eigenvalue decomposition. Standard Cholesky decomposition yields a upper triangular matrix in $\mathcal{O}(N^3)$ floating point operations the outcome of which can directly be used as $\sqrt{\mathbf{C}}^{\mathrm{T}}$. That is, the $\mathbf{s}$ vectors are obtained by matrix multiplication of the transposed outcome of the Cholesky algorithm with the standard normal vector $\mathbf{N}(\mathbf{0},\mathbf{I})$.

4 Comparison between CMSA-ES and CMA-ES

In order to demonstrate the effectiveness of the $\mathbf{C}$ adaptation rule (R7) in Fig. 2 and the choice of the parameters, empirical investigations are necessary to evaluate the behavior of the CMSA-ES and to compare it with the state-of-the-art $(\mu/\mu_W,\lambda)$-CMA-ES [13].

The CMSA-ES is a straightforward implementation of the algorithm in Fig. 2 using (2) and (1) for τ_{c} and τ, respectively. A truncation ratio of $\mu/\lambda = 1/4$ has been used throughout the simulations. This may be regarded as a compromise w.r.t. the progress rate under non-noisy conditions and final fitness error under additive symmetric fitness noise with constant strength (e.g. constant standard deviation) [8]. Furthermore, this choice is consonant with Hansen's recommendation to use "variance effective selection mass" $\mu_{\mathrm{eff}} = \lambda/4$ in the hybrid CMA-ES which transfers to $\mu = \lambda/4$ in the case of intermediate (uniformly weighted) recombination.

4.1 Test Functions

Tests have been performed on 12 test functions belonging to different problem classes. We will report results for four of them displayed in Tab. 1 each representing one class. The results of the other eight functions are qualitatively similar to these classes. We chose the sphere function as a kind of baseline for all continuous optimization tasks, the Schwefel ellipsoid because of the required adaptation of the covariance matrix and its special spectrum, the Rosenbrock function because it requires continuous change of the covariance matrix and the Rastrigin function because of its multi-modality.

Table 1. Test functions, initialization, and stop criterion for the evaluation of the CMA-ES

Name	Function	$\mathbf{y}_{\mathrm{init}}$	σ_{init}	f_{stop}
Sphere	$f_{\mathrm{Sp}}(\mathbf{y}) := \sum_{i=1}^{N} y_i^2$	$(1,\ldots,1)$	1	10^{-10}
Schwefel Ellipsoid	$f_{\mathrm{Sch}}(\mathbf{y}) := \sum_{i=1}^{N}\left(\sum_{j=1}^{i} y_i\right)^2$	$(1,\ldots,1)$	1	10^{-10}
Rosenbrock	$f_{\mathrm{Ros}}(\mathbf{y}) := \sum_{i=1}^{N-1}\left(100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2\right)$	$(0,\ldots,0)$	0.1	10^{-10}
Rastrigin	$f_{\mathrm{Ras}}(\mathbf{y}) := 10N + \sum_{i=1}^{N}\left(y_i^2 - 10\cos(2\pi y_i)\right)$	$\lVert\mathbf{y}\rVert = 10$	5	10^{-10}

Note, all test functions except Rastrigin's use a deterministic initialization for the object parameter vector $\mathbf{y}$. In the case of Rastrigin's function, the initial vector is randomly initialized on a hypersphere with given radius $\lVert\mathbf{y}\rVert$.

4.2 Simulation Settings

The simulation settings are directly taken from [13]. Both algorithms are compared for search space dimensionalities $N = 2, 3, 5, 10, 20, 40, 80,$ and 160 considering offspring populations sizes $\lambda = 8$, $\lambda = 4N$, and $\lambda = 4N^2$. For the latter population sizes, the maximum dimensionality of $N = 80$ has been chosen in order to keep the simulation time within reasonable limits. For each N-λ-combination, 20 independent runs have been used to obtain the average number of generations to reach f_{stop} (given in Tab. 1). These average generation numbers together with the corresponding standard deviation (displayed as error bars) vs. search space dimensionality N are displayed in the plots.

4.3 Results

The somewhat surprising results for the sphere function are presented in Fig. 3. Usually it is expected that the CMA-ES works better than self-adaptive ES on the sphere model due to the use of *cumulative step length* adaptation (CSA) in the CMA-ES [6]. Since both CMA and CMSA start with an initial covariance matrix $\mathbf{C} = \mathbf{I}$, i.e., with isotropic mutations, the superiority of CSA must be questioned. This is consonant with observations that the CMA-ES does not work well with population sizes $\lambda \gg N$. However, even more remarkable is the observation that the new *self-adaptive* CMSA-ES works comparably well in the small population and small search space dimensionality regime.

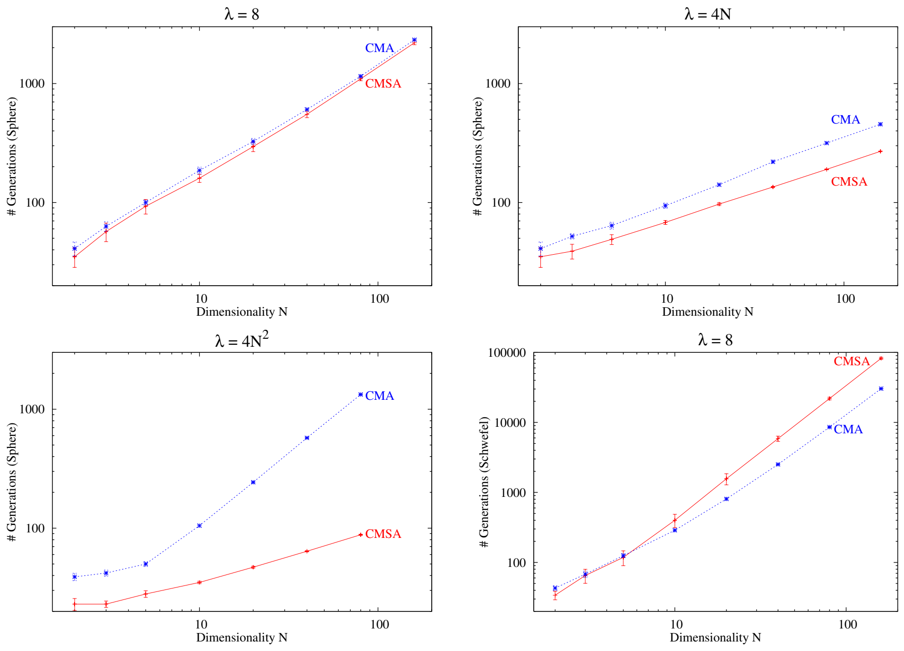


Fig. 3. Top row and bottom left: performance comparison on the sphere test function. Bottom right: performance comparison on Schwefel's Ellipsoid test function for constant $\lambda = 8$.

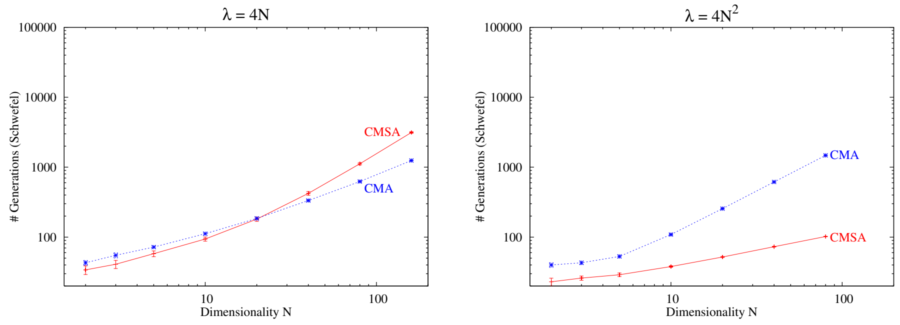


Fig. 4. Detailed performance comparison on Schwefel's Ellipsoid test function

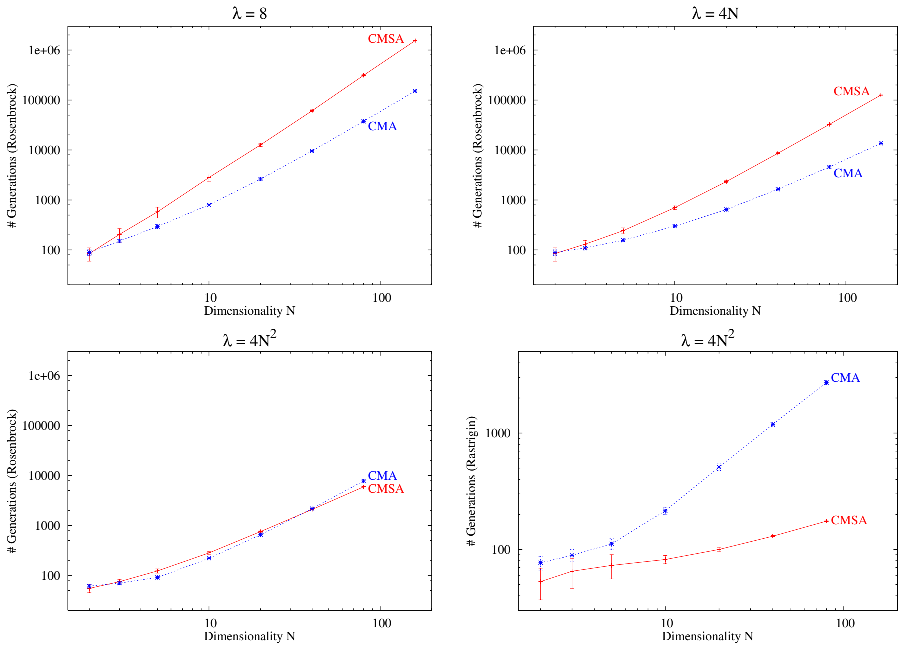


Fig. 5. Detailed performance comparison on Rosenbrock's test function and on Rastrigin test function (bottom right figure)

Originally, the CMA-ES and its recent hybrid versions were designed to adapt to arbitrary quadratic test functions. Therefore, the comparison of the performance on the ellipsoidal test function class provides a good basis to evaluate the different strategies. "Schwefel's Ellipsoid" is a rotated ellipsoid with moderately increasing eigenvalue spectrum (w.r.t. N), but an isolated largest eigenvalue. As can be seen in the left graph of Fig.4, the performance of the CMSA changes to the worse (compared to CMA) if

N gets larger. This is due to the increasing condition number of the mixing matrix in the ellipsoid function when N gets larger. However, as to large population sizes (right graph in Fig.4), CMSA performs better.

The Rosenbrock function seems to be somewhat harder for the CMSA-ES as can be seen in Fig. 5 in the case of constant and linear population sizing. In the case of quadratic population sizing both strategies perform nearly equally well. It seems that the path cumulation with decay rates proportional to $1/N$ (or larger) is a necessary ingredient in CMA-ES to effectively change the covariance matrix. This cannot be accomplished by the simple update rule (R7) used in our CMSA-ES when using population sizes of $\mathcal{O}(N)$.

For Rastrigin's function, only the quadratic population sizing has been used because the constant $\lambda = 8$ and the linear population sizing $\lambda = 4N$ does not ensure convergence to the global optimizer. It is to be mentioned that the quadratic population sizing $\lambda = 4N^2$ is *not* the optimal population sizing for this problem class. Actually, the optimal population sizing is weakly sublinear so that $\lambda \propto N$ would be the better choice. However, the proportionality factor is rather large. That is why, one observes convergence to local optima in runs with $\lambda = 4N^2$ for small N. This is also reflected in the larger standard deviations of the generation numbers in Fig. 5 (bottom right). As to the performance, one sees that the CMSA-ES clearly beats the CMA-ES. Similar behavior can be expected for other multi-modal test functions where the global optimizer is surrounded by a huge number of local optima.

5 Summary and Conclusion

In this paper, we have outlined the new $(\mu/\mu_I, \lambda)$-CMA-σ-SA-ES algorithm that uses mutative self-adaptation instead of cumulative step length adaptation to adjust the global step size σ during search. Compared to the standard CMA-ES which has (at least) *four* exogenous strategy parameters to be fixed, our new strategy contains only two, the time constants τ and τ_c. While the choice of some of those strategy parameters in CMA-ES is based on extensive empirical investigations, the new CMSA-ES time constants rely on information theoretical considerations.

The comparison of the CMSA-ES with the current state-of-the-art Evolution Strategy for real-valued parameter optimization, revealed a general pattern. While the CMA-ES performed slightly better for small population sizes, the newly proposed CMSA-ES achieved considerably better results for large population sizes. Surprisingly, for the sphere function both algorithms worked equally well even for small population sizes. Generally, we believe that due to its improved clarity and simplicity, the newly proposed algorithm is a serious competitor to the established CMA-ES. In case of large populations, we clearly recommend to employ the CMSA-ES. Large populations are required in particular in the context of robust optimization [8, 12]. Even for practical applications large populations can be feasible, e.g. in the context of massive parallelization or rapid serialization of experiments like in quantum control [14]. Therefore, the increased performance for larger population sizes of the proposed CMSA-ES has potential practical implications.

References

[1] Hansen, N., Ostermeier, A., Gawelczyk, A.: On the Adaptation of Arbitrary Normal Mutation Distributions in Evolution Strategies: The Generating Set Adaptation. In: Eshelman, L.J. (ed.) Proc. 6th Int'l Conf. on Genetic Algorithms, pp. 57–64. Morgan Kaufmann Publishers, Inc., San Francisco (1995)

[2] Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (2001)

[3] Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)

[4] Auger, A., Hansen, N.: A Restart CMA Evolution Strategy with Increasing Population Size. In: Congress on Evolutionary Computation, vol. 2, pp. 1769–1776. IEEE, Los Alamitos (2005)

[5] Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical Report, Nanyang Tech. University, Singapore (2005)

[6] Beyer, H.-G., Arnold, D.V.: Qualms Regarding the Optimality of Cumulative Path Length Control in CSA/CMA-Evolution Strategies. Evolutionary Computation 11(1), 19–28 (2003)

[7] Beyer, H.-G., Olhofer, M., Sendhoff, B.: On the Behavior of $(\mu/\mu_I, \lambda)$-ES Optimizing Functions Disturbed by Generalized Noise. In: De Jong, K., Poli, R., Rowe, J. (eds.) Foundations of Genetic Algorithms, 7, pp. 307–328. Morgan Kaufmann, San Francisco (2003)

[8] Beyer, H.-G., Sendhoff, B.: Evolution Strategies for Robust Optimization. In: Congress on Evolutionary Computation (CEC), pp. 1346–1353. IEEE Press, Los Alamitos (2006)

[9] Auger, A., Schoenauer, M., Vanhaecke, N.: LS-CMA-ES: A Second-Order Algorithm for Covariance Matrix Adaptation. In: Yao, X., et al. (eds.) Parallel Problem Solving from Nature 8, pp. 182–191. Springer, Berlin (2004)

[10] Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-Objective Optimization. Evolutionary Computation 15(1), 1–28 (2007)

[11] Arnold, D.V., Beyer, H.-G.: Performance Analysis of Evolutionary Optimization with Cumulative Step Length Adaptation. IEEE Transactions on Automatic Control 49(4), 617–622 (2004)

[12] Beyer, H.-G., Sendhoff, B.: Robust Optimization - A Comprehensive Survey. Computer Methods in Applied Mechanics and Engineering 196(33–34), 3190–3218 (2007)

[13] Hansen, N., Kern, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In: Yao, X., et al. (eds.) Parallel Problem Solving from Nature 8, pp. 282–291. Springer, Berlin (2004)

[14] Amstrup, B., Tóth, G.J., Szabó, G., Rabitz, H., Lörincz, A.: Genetic Algorithm with Migration on Topology Conserving Maps for Optimal Control of Quantum Systems. J. Phys. Chem. 99, 5206–5213 (1995)

Enhancing the Performance of Maximum–Likelihood Gaussian EDAs Using Anticipated Mean Shift

Peter A.N. Bosman[1], Jörn Grahl[2], and Dirk Thierens[3]

[1] Centre for Mathematics and Computer Science, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Peter.Bosman@cwi.nl

[2] University of Mannheim, Mannheim, Germany
joern.grahl@bwl.uni-mannheim.de

[3] Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
Dirk.Thierens@cs.uu.nl

Abstract. Many Estimation–of–Distribution Algorithms use maximum-likelihood (ML) estimates. For discrete variables this has met with great success. For continuous variables the use of ML estimates for the normal distribution does not directly lead to successful optimization in most landscapes. It was previously found that an important reason for this is the premature shrinking of the variance at an exponential rate. Remedies were subsequently successfully formulated (i.e. Adaptive Variance Scaling (AVS) and Standard–Deviation Ratio triggering (SDR)). Here we focus on a second source of inefficiency that is not removed by existing remedies. We then provide a simple, but effective technique called Anticipated Mean Shift (AMS) that removes this inefficiency.

1 Introduction

Estimation–of–Distribution Algorithm (EDAs) are a specific type of Evolutionary Algorithm (EA). EDAs are characterized by the way in which new solutions are generated. The information in all selected solutions is combined at once. To this end, an interim representation that compresses and summarizes this information is used: a probability distribution over the solution space. New solutions are generated by sampling the distribution.

Efficient optimization is guaranteed under suitable conditions [14]. In practice it is however impossible to meet these conditions because arbitrarily complex distributions are required. Hence, practical techniques are required. In this paper, we focus on optimization of numerical functions using continuous distributions. The use of the normal distribution or combinations thereof is the most commonly adopted choice. It has already been so since the first EDAs in continuous spaces were introduced [4,11,17,18]. An important question is how efficient EDAs are in the continuous domain using such practical distributions.

Recently, it was shown that without precaution, premature convergence is likely to occur with these approaches, even on slope–like regions of the search space [7,8,9]. The main reason for this is that the variance decreases too fast at an exponential rate. The current state of the art exists of techniques that attempt to remedy premature convergence (e.g. adaptive variance scaling [2,8,15]). Here we show that another source of inefficiency however exists that cannot be removed by these remedies. The use of ML estimates results in a distribution that describes the set of selected solutions well. On a slope however, it is not the set of selected solutions that is interesting, but it is the direction of descent. Efficient sampling along the direction of descent is therefore not guaranteed, even if the covariance matrix is scaled. We shall illustrate this problem further in this paper and present a remedy that we call AMS (Anticipated Mean Shift). The use of AMS improves performance, even if no covariances are estimated. Also, the resulting EDA still only uses ML estimates, which are a well–understood and sensible way of estimating parameters from data. We call the new EDA AMaLGaM–IDEA (Adapted Maximum–Likelihood Gaussian Model — Iterated Density–Estimation Evolutionary Algorithm) or just AMaLGaM for short. We compare the results of AMaLGaM with CMA–ES, currently the most efficient evolution strategy for continuous optimization.

2 Maximum–Likelihood Estimations, AVS and SDR

2.1 Maximum–Likelihood Estimations

We introduce a random variable X_i for each real–valued problem variable x_i, $i \in \{0, 1, \ldots, l-1\}$ where l is the problem dimensionality. The normal distribution $P^{\mathcal{N}}_{(\boldsymbol{\mu_v}, \boldsymbol{\Sigma^v})}(X_{\boldsymbol{v}})$ for a vector of random variables $X_{\boldsymbol{v}} = (X_{v_0}, X_{v_1}, \ldots, X_{v_{|\boldsymbol{v}|-1}})$ is parametrized by a vector $\boldsymbol{\mu_v}$ of means and a symmetric covariance matrix $\boldsymbol{\Sigma^v}$:

$$P^{\mathcal{N}}_{(\boldsymbol{\mu_v}, \boldsymbol{\Sigma^v})}(X_{\boldsymbol{v}} = \boldsymbol{x}) = \frac{(2\pi)^{-\frac{|\boldsymbol{v}|}{2}}}{(\det \boldsymbol{\Sigma^v})^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_v})^T (\boldsymbol{\Sigma^v})^{-1}(\boldsymbol{x}-\boldsymbol{\mu_v})} \tag{1}$$

In an EDA, the distribution parameters are estimated from the vector of selected solutions $\boldsymbol{\mathcal{S}}$. Maximum–likelihood (ML) estimation is a principled and commonly–adopted approach. ML estimates for the mean and covariance matrix are given by the sample average and sample covariance matrix respectively:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{v}} = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} (\boldsymbol{\mathcal{S}}_j)_{\boldsymbol{v}} \qquad \hat{\boldsymbol{\Sigma}}^{\boldsymbol{v}} = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} ((\boldsymbol{\mathcal{S}}_j)_{\boldsymbol{v}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{v}})((\boldsymbol{\mathcal{S}}_j)_{\boldsymbol{v}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{v}})^T \tag{2}$$

To reduce the effort in learning the joint distribution, factorizations are commonly used. A factorization factors the joint distribution into a product of smaller joint (possibly conditional) distributions [12]. Learning effort is reduced the most using the well–known univariate factorization in which all variables are independent, i.e. the distribution is written as $\prod_{i=0}^{l-1} P(X_i)$. In the case of the normal distribution this means that all covariances are zero. Allowing for all possible dependencies implies use of the full covariance matrix. As an intermediate choice, a greedy algorithm can be used to determine and use only the most important dependencies. To this end, Bayesian factorizations are typically used in EDAs [13,16]. To briefly recall Bayesian factorizations, recall that the vector of random variables indicated by $X_{\boldsymbol{\pi}_i}$ on which X_i is conditioned is called the vector of parents of X_i and that the distribution is written $\prod_{i=0}^{l-1} P(X_i|X_{\boldsymbol{\pi}_i})$. Let $\boldsymbol{W^j}$ be the inverse of the symmetric covariance matrix, i.e. $\boldsymbol{W^j} = (\boldsymbol{\Sigma^j})^{-1}$. ML estimates of $P^{\mathcal{N}}(X_i|X_{\boldsymbol{\pi}_i})$ can be expressed in terms of Equation 2 [4]:

$$\hat{P}^{\mathcal{N}}(X_i{=}x_i \mid X_{\boldsymbol{\pi}_i}{=}x_{\boldsymbol{\pi}_i}) = \frac{1}{(\breve{\sigma}_i\sqrt{2\pi})}e^{\frac{-(x_i-\breve{\mu}_i)^2}{2\breve{\sigma}_i^2}} \tag{3}$$

$$\text{where} \begin{cases} \breve{\sigma}_i = \frac{1}{\sqrt{\hat{\boldsymbol{W}}_{00}^{(i,\boldsymbol{\pi}_i)}}} \\ \breve{\mu}_i = \frac{\hat{\boldsymbol{\mu}}_i\hat{\boldsymbol{W}}_{00}^{(i,\boldsymbol{\pi}_i)} - \sum_{j=0}^{|\boldsymbol{\pi}_i|-1}(x_{(\boldsymbol{\pi}_i)_j} - \hat{\boldsymbol{\mu}}_{(\boldsymbol{\pi}_i)_j})\hat{\boldsymbol{W}}_{(j+1)0}^{(i,\boldsymbol{\pi}_i)}}{\hat{\boldsymbol{W}}_{00}^{(i,\boldsymbol{\pi}_i)}} \end{cases}$$

Because Equation 3 has the form of a single–dimensional normal distribution, sampling from the Bayesian factorization is again straightforward once all relevant computations have been performed. Depending on the independencies expressed by the factorization, the density ellipsoids can be aligned with any axis. Use of the complete covariance matrix corresponds to a Bayesian factorization in which each X_i is conditioned on all X_j with $j > i$.

The full covariance matrix requires the most data to learn properly because all covariances need to be estimated. Although this argument advocates the univariate factorization, the use of it in an EDA brings about important limitations. The ellipsoid–shaped density contours can only be aligned with the main axes. This means that a function such as the Ellipsoid function ($\sum_{i=0}^{l-1} 10^{6\frac{i}{l-1}} x_i^2$) can be optimized efficiently. However, a rotated version of the same function introduces strong dependencies between the variables because each quadratic form is scaled differently. The contours of the function can no longer be matched by the contours of the univariately factorized normal distribution and optimization fails. Hence, a full covariance matrix is required to ensure rotation–invariance [10].

2.2 AVS

To remedy the problem of the prematurely vanishing variance, the variance can be scaled beyond its ML estimate [15]. One successful scheme for doing so is called adaptive variance scaling (AVS) [8]. This scheme allows the EDA to solve problems that it couldn't solve without scaling the variance.

In AVS, a variance multiplier c^{AVS} is maintained. For sampling, $c^{\text{AVS}}\hat{\boldsymbol{\Sigma}}$ is used instead of $\hat{\boldsymbol{\Sigma}}$. If the best fitness value improves, then the current size of the variance allows for progress. Hence, a further enlargement of the variance may allow further improvement in the next generation. To fight the variance–diminishing effect of selection, c^{AVS} is scaled by $\eta^{\text{INC}} > 1$. If there is no improvement, the exploration range may be too large and c^{AVS}. is decreased by a factor $\eta^{\text{DEC}} \in [0, 1]$. For symmetry, $\eta^{\text{DEC}} = 1/\eta^{\text{INC}}$. As the objective of the AVS scheme is to enlarge the variance to prevent premature convergence, $c^{\text{AVS}} \geq 1$ is enforced.

2.3 SDR

With AVS, improvements increase c^{AVS}. If the mean is already near the optimum, no further variance enlargement is necessary however. Let $\overline{\boldsymbol{x}^{\text{IMP}}}(t)$ denote the average of improvements in generation t. Further enlargement of c^{AVS} in generation $t+1$ is triggered whenever $\overline{\boldsymbol{x}^{\text{IMP}}}(t)$ lies further away from $\hat{\boldsymbol{\mu}}(t)$ than a single standard deviation. To this end, the standard–deviation ratio (SDR) needs to be computed. The SDR is the ratio a/b of the distance to the mean of a) $\overline{\boldsymbol{x}^{\text{IMP},i}}(t)$ and b) the contour line of one standard deviation in the same direction. The SDR is independent of the sample range and has a fixed, predefined notion of being "close" to the mean [2].

3 Anticipated Mean Shift

3.1 Motivation

Most EDAs have been benchmarked using initialization ranges (IRs) centered around the optimum. An EDA based on the normal distribution with ML estimates focuses its search by contracting the region of exploration towards the mean. Hence, problems and the search bias of the EDA are favorably matched, leading to possibly overenthousiastic conclusions. This is already known to be the case for other contractive operators such as intermediate recombination [5]. Hence, it is important to specifically investigate the non–symmetric case.

A simple opposite of a symmetric function is the linear slope. Previous research focused on the one–dimensional case [2,9]. Here, we consider two dimensions, i.e. $f(\boldsymbol{x}) = x_0 + x_1$. Use of the univariate factorization on this problem corresponds to the same situation of a single dimension studied earlier. We therefore focus on the case in which covariances are estimated also. The direction $\boldsymbol{u}$ of steepest descent obeys $u_0 = u_1$ and $u_i \leq 0$. Thus, it is most efficient to have the density ellipsoids parallel to and elongated along the line $x_0 = x_1$. Conversely, the worst alignment is parallel to and elongated along $x_0 = -x_1$.

Figure 1 shows the density contours in the case of the full covariance matrix for the first six subsequent generations. The density contours shown are the 95% error ellipses. When ML estimates are used only, the normal distribution quickly contracts. Initially, the population is spread uniformly in a square. On a two–dimensional slope the selected solutions form a triangle. Fitting a normal distribution with ML results in density contours aligned in the worst way. Scaling the covariance matrix almost solely increases search effort in the futile direction perpendicular to the best direction.

This effect was first noted in [19]. The same study proposed a first remedy. The remedy employs minimization of cross–entropy in which both the selected solutions and the population are used. Although the problem at hand was alleviated by this remedy, the resulting scaling behavior was reported in that same study to be inferior to AVS when symmetric initialization is used. Also, the well–known ML estimates can no longer be used. Here, we provide a simple, yet elegant and intuitive alternative way to overcome the inefficiency at hand that

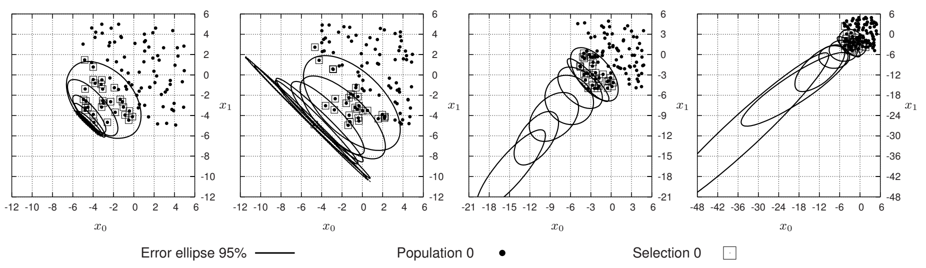


Fig. 1. Estimated normal distribution in the first 6 generations of typical runs with (from left to right): ML estimates, SDR–AVS, AMS and AMaLGaM on the two–dimensional slope $f(\boldsymbol{x}) = x_0 + x_1$ with IR $[-5; 5] \times [-5; 5]$. The density contours are the 95% error ellipses. Also shown are the population and selection in generation 0.

ultimately leads even to improvements over the use of SDR–AVS alone in the case of symmetric initialization.

3.2 Technique

The difference of the means in two subsequent generations indicates the direction in which the solutions are moved to obtain better fitness. Let $\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$ denote for generation t the mean shift for generations $t-1$ and t:

$$\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) = \hat{\boldsymbol{\mu}}(t) - \hat{\boldsymbol{\mu}}(t-1) \tag{4}$$

Note that our definition of mean shift differs from the one used in the mean–shift clustering algorithm that was studied in relation to EDAs elsewhere [6]. A straightforward anticipation of the mean shift that is required to obtain further improvements in generation $t+1$ is $\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$. It is therefore sensible to alter $100\alpha\%$ of all newly sampled solutions $\boldsymbol{x}$ in generation t by moving them a certain fraction δ in the direction of the previously observed the mean shift, i.e.:

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \delta \hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \tag{5}$$

We call this operation Anticipated Mean Shift (AMS).

When centered over an optimum, $\hat{\boldsymbol{\mu}}(t) \approx \hat{\boldsymbol{\mu}}(t-1)$ and therefore $\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \approx \boldsymbol{0}$, leaving the original approach unchanged. On a slope, AMS causes an important adjustment of $\hat{\boldsymbol{\Sigma}}$ that is estimated still using only ML. Solutions are selected from three sets: I) previously selected solutions (i.e. elitist solutions), II) new solutions *without* AMS and III) new solutions *with* AMS. Since set II is generated from a model that was estimated with ML from set I, these two sets share a similar region. Set III is further down the slope. If selection now selects solutions from both regions, the density contours are re–aligned, see Figure 1. Note that if the mean is nearing a peak and AMS overshoots the optimum, the mean shift in the next generation will be much smaller because the mean shift will be caused again mostly by the non–anticipated solutions. This thus resets the approach.

Number of adaptations (setting $\boldsymbol{\alpha}$). We assume that the best τn solutions are selected, where n is the population size. Moreover, the selected solutions survive and $(1-\tau)n$ new solutions are generated to refill the population.

On a slope, all of the $\alpha(1-\tau)n$ altered solutions will be better and get selected. Now, if $\tau \geq \alpha$ only the altered solutions are selected, leaving the orientation of the density contours unchanged. For a change to occur, the selected solutions must consist of both unaltered and altered solutions. Ideally, these proportions are equally sized, which gives $\alpha(1-\tau)n = \frac{1}{2}\tau n$ and thus $\alpha = \frac{\tau}{2-2\tau}$. As using information about the anticipated mean shift is still only predictive, we want to alter no more than 50% of the newly sampled solutions, i.e. $\alpha \leq 0.5$. This restricts the selection percentile: $\alpha \leq 0.5 \quad \Leftrightarrow \quad \frac{\tau}{2-2\tau} \leq 0.5 \quad \Leftrightarrow \quad \tau \leq 0.5$.

Adaptation length (setting $\boldsymbol{\delta}$). On a slope, set III in generation t constitutes 50% of the selected solutions in generation $t+1$. The other 50% comes from sets I and II. The mean of the latter two sets is $\hat{\boldsymbol{\mu}}(t)$. The mean of set III is $\hat{\boldsymbol{\mu}}(t) + \delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$. For the suggested value of α, the mean of the selected set in generation $t+1$ is[1] $\hat{\boldsymbol{\mu}}(t+1) = \frac{1}{2}\left(\hat{\boldsymbol{\mu}}(t) + \hat{\boldsymbol{\mu}}(t) + \delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)\right) = \hat{\boldsymbol{\mu}}(t) + \frac{\delta}{2}\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$. The mean shift in generation $t+1$ is then $\hat{\boldsymbol{\mu}}^{\text{Shift}}(t+1) = \hat{\boldsymbol{\mu}}(t+1) - \hat{\boldsymbol{\mu}}(t) = \frac{\delta}{2}\hat{\boldsymbol{\mu}}^{\text{Shift}}(t)$. Hence, for any $\delta < 2$ the mean shift is expected to become smaller. Because the newly estimated mean falls in between the two sets, an ML estimate captures also the variance *between* the two sets. This causes the density to be aligned more favorably with the direction of descent. With repetition, the re–aligned density can result in a larger mean–shift. Hence a value of $\delta = 2$ suffices. The illustrations in Figure 1 were obtained using $\delta = 2$.

4 Combining SDR, AVS and AMS: AMaLGaM

On a slope it makes sense to accelerate the search. The AVS scheme provides a principled way to achieve this. If improvements occur far away from the mean in subsequent generations, c^{AVS} is enlarged. This relation between c^{AVS} and improvements allows c^{AVS} to be seen as a general accelerator. We therefore rename the variance multiplier c^{AVS} to distribution multiplier $c^{\text{Multiplier}}$. Not only do we use $c^{\text{Multiplier}}\hat{\boldsymbol{\Sigma}}$ instead of $\hat{\boldsymbol{\Sigma}}$ upon sampling the distribution, we also use

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + c^{\text{Multiplier}}\delta\hat{\boldsymbol{\mu}}^{\text{Shift}}(t) \tag{6}$$

upon applying AMS. This accelerates descent on a slope. In Figure 1 the effect of combining AVS with AMS can be seen when traversing the slope in two dimensions. The distribution gets rotated and elongated along the direction of improvement much faster than without the use of the distribution multiplier (note the difference in scale on both axes).

The combination of SDR, AVS and AMS adaptively changes both the covariance matrix and the mean–shift. It prevents premature convergence due to inefficient sampling that results from fitting only the set of selected solutions without considering the direction of descent. We name this composite AMS–SDR–AVS technique

[1] Equality only holds for an infinite population size, it is an approximation otherwise.

AMaLGaM (Adapted Maximum–Likelihood Gaussian Model). Pseudo–code may be found in a technical report [3].

5 Guidelines and Comparison with CMA–ES

It is important to compare results with literature. It is equally important to have guidelines to use in subsequent applications and research. We therefore first derive guidelines and then use them to compare AMaLGaM with CMA–ES, currently the most efficient evolution strategy for continuous optimization.

5.1 Guidelines

To derive guidelines, we use 10 benchmark functions to be minimized taken from literature [8,10]. A function is considered to be optimized if the best solution has reached a certain value–to–reach (VTR). The VTR for all functions except the ridge functions is 10^{-10}. For the two ridge functions the VTR is -10^{10}.

Name	**Definition**
Sphere	$\sum_{i=0}^{l-1} x_i^2$
Ellipsoid	$\sum_{i=0}^{l-1} 10^{6\frac{i}{l-1}} x_i^2$
Cigar	$x_0^2 + \sum_{i=1}^{l-1} 10^6 x_i^2$
Tablet	$10^6 x_1^2 + \sum_{i=1}^{l-1} x_i^2$
Cigar Tablet	$x_0^2 + \sum_{i=1}^{l-2} 10^4 x_i^2 + 10^8 x_{l-1}^2$

Name	**Definition**
Two Axes	$\sum_{i=0}^{\lfloor l/2 \rfloor -1} 10^6 x_i^2 + \sum_{i=\lfloor l/2 \rfloor -1}^{l-1} x_i^2$
Different Powers	$\sum_{i=0}^{l-1} \lvert x_i \rvert^{2+10\frac{i}{l-1}}$
Rosenbrock	$\sum_{i=0}^{l-2} \left(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$
Parabolic Ridge	$-x_1 + 100 \sum_{i=1}^{l-1} x_i^2$
Sharp Ridge	$-x_1 + 100 \sqrt{\sum_{i=1}^{l-1} x_i^2}$

We determined the optimal population size for AMaLGaM in the naive variant (i.e. univariate factorization), the learning variant (i.e. Bayesian factorization) and the full covariance matrix variant (i.e. unfactorized). For the full covariance matrix variant we used the functions as provided above as well as their rotated variants. With rotation each pair of variables in a solution is rotated 45 degrees before function evaluation takes place (for more details, see [3]).

IRs of $[-7.5; 7.5]$ (symmetric around optimum), $[-10, 5]$ (asymmetric) and $[-115, -100]$ (far–away) were used. We combined all scalability plots and determined on the basis thereof a guideline for the population size. For each variant, a minimal population size of 20 for $l = 1$ was determined. The guidelines and the combined scalability plots are presented in Figure 2.

Because AMaLGaM solves problems that can't be solved if only SDR–AVS or ML–estimates are used, no comparison is presented here with SDR–AVS or ML–estimates. It was found though, that AMaLGaM requires on average 0.67 times the evaluations of SDR–AVS. Hence, not only does AMaLGaM enlarge the class of problems that can be solved by the EDA, it also improves its efficiency. We also note that using the full covariance matrix no significant difference could be detected in solving rotated and unrotated versions of the problems. Hence, AMaLGaM can be said to be robust to rotations of the search space. More details on these additional results may be found in the technical report [3].

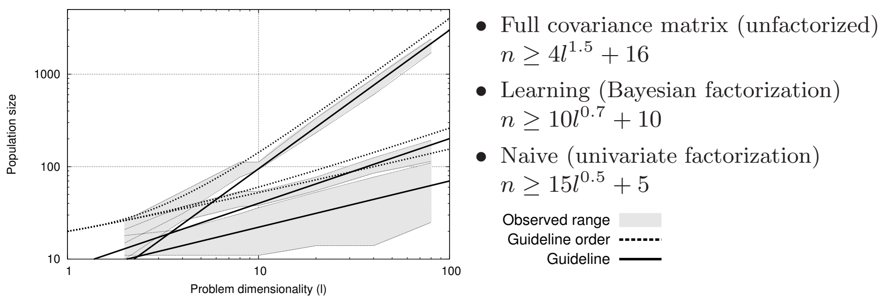


Fig. 2. Observed and guideline population size that leads to the minimum number of evaluations for AMaLGaM to reach the VTR, averaged over 100 independent runs. The gray areas are the observed population sizes for all problems.

5.2 Comparisons

We used the guidelines defined in Section 5.1 and ran AMaLGaM 100 independent times on each of the benchmark problems. For the parameter settings of the CMA–ES, we used the guidelines provided in the literature also [10]. To prevent biased results from symmetric initialization, we used the far–away IR.

Scalability. We computed a least–squares fit to $\alpha l^{\beta} + \gamma$ for the number of required evaluations. The fit was always found to be highly accurate. The results are summarized in Figure 3.

Comparing naive, learning and full covariance matrix. The naive variant scales better than the Bayesian variant, which in turn scales better than the variant that uses the full covariance matrix. However, this only holds for functions

Function	Algorithm	β	α	γ
Sphere	AMaLGaM–N	1.23	$2.74{\cdot}10^2$	$9.46{\cdot}10^0$
	AMaLGaM–L	1.34	$2.46{\cdot}10^2$	$1.63{\cdot}10^2$
	AMaLGaM–F	2.05	$1.09{\cdot}10^2$	$4.08{\cdot}10^2$
	CMA–ES	0.94	$2.38{\cdot}10^2$	$3.17{\cdot}10^2$
Ellipsoid	AMaLGaM–N	1.24	$3.33{\cdot}10^2$	$8.20{\cdot}10^0$
	AMaLGaM–L	1.36	$2.90{\cdot}10^2$	$1.31{\cdot}10^2$
	AMaLGaM–F	2.09	$1.14{\cdot}10^2$	$4.87{\cdot}10^2$
	CMA–ES	1.92	$6.40{\cdot}10^1$	$1.79{\cdot}10^3$
Cigar	AMaLGaM–N	1.25	$3.40{\cdot}10^2$	$-2.30{\cdot}10^1$
	AMaLGaM–L	1.35	$3.20{\cdot}10^2$	$4.48{\cdot}10^1$
	AMaLGaM–F	2.08	$1.30{\cdot}10^2$	$4.14{\cdot}10^2$
	CMA–ES	0.90	$7.18{\cdot}10^2$	$-2.16{\cdot}10^2$
Tablet	AMaLGaM–N	1.22	$2.95{\cdot}10^2$	$1.12{\cdot}10^2$
	AMaLGaM–L	1.32	$2.77{\cdot}10^2$	$1.80{\cdot}10^2$
	AMaLGaM–F	2.04	$1.13{\cdot}10^2$	$5.24{\cdot}10^2$
	CMA–ES	1.64	$1.17{\cdot}10^2$	$1.59{\cdot}10^3$
Cigar tablet	AMaLGaM–N	1.22	$3.54{\cdot}10^2$	-4.14e-01
	AMaLGaM–L	1.34	$3.21{\cdot}10^2$	$8.52{\cdot}10^1$
	AMaLGaM–F	2.07	$1.23{\cdot}10^2$	$4.93{\cdot}10^2$
	CMA–ES	1.40	$2.16{\cdot}10^2$	$1.48{\cdot}10^3$

Function	Algorithm	β	α	γ
Two axes	AMaLGaM–N	1.27	$3.06{\cdot}10^2$	$4.62{\cdot}10^1$
	AMaLGaM–L	1.37	$2.86{\cdot}10^2$	$1.18{\cdot}10^2$
	AMaLGaM–F	2.10	$1.11{\cdot}10^2$	$5.08{\cdot}10^2$
	CMA–ES	2.00	$7.91{\cdot}10^1$	$1.68{\cdot}10^3$
Different powers	AMaLGaM–N	1.39	$1.49{\cdot}10^2$	$1.98{\cdot}10^2$
	AMaLGaM–L	1.41	$1.70{\cdot}10^2$	$1.94{\cdot}10^2$
	AMaLGaM–F	2.09	$7.75{\cdot}10^1$	$3.78{\cdot}10^2$
	CMA–ES	1.65	$1.55{\cdot}10^2$	$1.14{\cdot}10^3$
Rosenbrock	AMaLGaM–N	1.55	$5.94{\cdot}10^3$	$-7.59{\cdot}10^3$
	AMaLGaM–L	1.70	$2.42{\cdot}10^2$	$1.43{\cdot}10^3$
	AMaLGaM–F	2.57	$5.58{\cdot}10^1$	$2.35{\cdot}10^3$
	CMA–ES	1.92	$7.25{\cdot}10^1$	$2.52{\cdot}10^3$
Parabolic ridge	AMaLGaM–N	1.02	$2.00{\cdot}10^2$	$1.57{\cdot}10^2$
	AMaLGaM–L	1.13	$2.75{\cdot}10^2$	$1.14{\cdot}10^2$
	AMaLGaM–F	2.01	$1.06{\cdot}10^2$	$3.38{\cdot}10^2$
	CMA–ES	1.01	$4.29{\cdot}10^2$	$3.43{\cdot}10^2$
Sharp ridge	AMaLGaM–N	0.95	$1.70{\cdot}10^2$	$2.02{\cdot}10^2$
	AMaLGaM–L	1.08	$1.57{\cdot}10^2$	$2.20{\cdot}10^2$
	AMaLGaM–F	1.87	$7.33{\cdot}10^1$	$3.35{\cdot}10^2$
	CMA–ES	0.78	$2.80{\cdot}10^3$	$-9.00{\cdot}10^3$

Fig. 3. Scalability regression coefficients on all benchmark problems averaged over 100 independent runs using the guidelines. The IR is $[-115, -100]$ for each variable.

that fit the model used. The naive method for instance cannot solve problems with many dependencies (e.g. rotated versions of the benchmark problems).

In additional experiments (for details, see the technical report [3]) it was found that the scalability of AMaLGaM does not change significantly when moving from asymmetric initialization to far–away initialization. This leads to the conclusion that AMaLGaM is also robust to translations, a property that earlier EDAs with ML estimates and even adaptive variance scaling do not have.

Comparing AMaLGaM and CMA. The scalability of CMA–ES ranges between the different variants of AMaLGaM. For some functions (e.g. Sphere), CMA–ES has a better scalability than even the naive variant of AMaLGaM. For other functions (e.g. Two axes) it has a scalability similar to the full variant of AMaLGaM. The scalability results of AMaLGaM are less variable, causing CMA–ES to be better on some functions and AMaLGaM to be better on other functions. CMA–ES has the upper hand in the comparison, especially if rotation invariance is desired. This requires use of the full covariance matrix. AMaLGaM then however has a scalability that is at most similar (e.g. Two axes).

Runtime. The number of required evaluations is important, especially if evaluations are time–consuming. The overall running time is however also important. With higher model complexity comes a larger learning and sampling time. Use of the full covariance matrix requires $\mathcal{O}(l^3)$ time. Assuming bounded complexity for the Bayesian network, the same asymptotic bound holds for the learning case with the commonly used greedy algorithm [4,16]. Hence, room for improvement exists to increase benefits from learning over using the full covariance matrix. Modelling time for the univariate factorization is only $\mathcal{O}(l)$. Detailed run–times per benchmark function and per algorithm are given in the technical report [3].

6 Summary, Discussion and Future Work

Using maximum–likelihood (ML) estimates for the normal distribution in an EDA, premature convergence is likely to occur. Optimization is only performed properly if the initialization range brackets the optimum. Optimization then mainly proceeds by contraction. Methods of adaptive variance scaling (AVS) provide a way to control the rate of contraction and turn it into expansion. Because ML estimates shape the density similar to the configuration of the selected solutions, the density contours can however be misaligned with the direction of descent. The variance then needs to be scaled to excessively large values to still make progress. We have proposed a simple, yet effective approach called anticipated mean shift (AMS) that removes this inefficiency. AMS advances sampled solutions in the direction of the mean shift of the previous generation. We analyzed this technique and provided rational settings for its parameters. We called the resulting EDA Adapted Maximum–Likelihood Gaussian Model — Iterated Density–Estimation Evolutionary Algorithm (AMaLGaM–I𝔻𝔼A or AMaLGaM for short). An experimental scalability analysis showed that AMaLGaM is robust to rotations and translations of the search space and is competitive with

CMA–ES under certain conditions. AMaLGaM therefore makes an important step in the progression of continuous EDAs for numerical optimization.

Adaptivity in real–valued optimization has long been acknowledged to be important [1]. Its use in ES has led to the development of CMA–ES. Both AMaLGaM and CMA–ES adapt a Gaussian model using various techniques. The view upon the Gaussian model is different however. In CMA–ES directions are modelled and thus the Gaussian mainly serves as a mutation operator. In EDAs the region of interest is directly modelled and thus the Gaussian mainly serves as a recombination operator. The type of adaptation required is therefore different. It is important to research and take note of results along both lines.

The practical applicability of AMaLGaM and CMA–ES depends on the problem dimensionality. Using the full covariance matrix, only problems of relatively small dimensionality can be tackled due to the high required computing time. This leaves only methods that consider a few dependencies or no dependencies at all (i.e. the naive AMaLGaM). Certainly, if there are many strong dependencies in the problem, the algorithm can't find the optimum. Still, due to its simplicity, speed, and effectiveness the naive AMaLGaM can well serve as a baseline EDA to be used for future comparison and for applications with many variables.

One important direction of future work that we are currently pursuing is a reduction of the required population size. To ensure the full covariance matrix is well–conditioned for inversion, the required population size is quite large. This requires many samples in the generation–wise ML estimate. CMA–ES on the other hand convolutes the covariance matrix over multiple generations. This reduces the required population size and directly leads to less function evaluations.

References

1. Beyer, H.-G., Deb, K.: On self–adaptive features in real–parameter evolutionary algorithms. IEEE Transactions on Evolutionary Computation 5(3), 250–270 (2001)
2. Bosman, P.A.N., Grahl, J., Rothlauf, F.: SDR: A better trigger for adaptive variance scaling in normal EDAs. In: Thierens, D., et al. (eds.) Proc. of the Genetic and Evol. Comp. Conf. — GECCO–2007, pp. 492–499. ACM Press, New York (2007)
3. Bosman, P.A.N., Grahl, J., Thierens, D.: Adapted maximum–likelihood Gaussian models for numerical optimization with continuous EDAs. CWI technical report SEN–E0704 (2007)
4. Bosman, P.A.N., Thierens, D.: Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 767–776. Springer, Heidelberg (2000)
5. Fogel, D.-B., Beyer, H.-G.: A note on the empirical evaluation of intermediate recombination. Evolutionary Computation 3(4), 491–495 (1996)
6. Gallagher, M., Frean, M.: Population–based continuous optimization, probabilistic modelling and mean shift. Evolutionary Computation 13(1), 29–42 (2005)
7. González, C., Lozano, J.A., Larrañaga, P.: Mathematical modelling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions 31(3), 313–340 (2002)

8. Grahl, J., Bosman, P.A.N., Rothlauf, F.: The correlation–triggered adaptive variance scaling IDEA. In: Keijzer, M., et al. (eds.) Proc. of the Genetic and Evol. Comp. Conf. — GECCO–2006, pp. 397–404. ACM Press, New York (2006)
9. Grahl, J., Minner, S., Rothlauf, F.: Behaviour of UMDAc with truncation selection on monotonous functions. In: Corne, D., et al. (eds.) Proceedings of the IEEE Congress on Evol. Comp. — CEC–2005, pp. 2553–2559. IEEE Computer Society Press, Los Alamitos (2005)
10. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA–ES). Evolutionary Computation 11(1), 1–18 (2003)
11. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.: Optimization in continuous domains by learning and simulation of Gaussian networks. In: Pelikan, M., et al. (eds.) Proc. of the OBUPM Workshop at the Genetic and Evol. Comp. Conf. — GECCO–2000, pp. 201–204. Morgan Kaufmann, San Francisco (2000)
12. Lauritzen, S.L.: Graphical Models. Clarendon Press (1996)
13. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E.: Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms (2006)
14. Mühlenbein, H., Höns, R.: The estimation of distributions and the minimum relative entropy principle. Evolutionary Computation 13(1), 1–27 (2005)
15. Ocenasek, J., Kern, S., Hansen, N., Müller, S., Koumoutsakos, P.: A mixed Bayesian optimization algorithm with variance adaptation. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 352–361. Springer, Heidelberg (2004)
16. Pelikan, M., Sastry, K., Cantú-Paz, E.: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications. Springer, Heidelberg (2006)
17. Rudlof, S., Köppen, M.: Stochastic hill climbing with learning by vectors of normal distributions. In: Furuhashi, T. (ed.) Proceedings of the First Online Workshop on Soft Computing — WSC1, pp. 60–70. Nagoya Univ. (1996)
18. Sebag, M., Ducoulombier, A.: Extending population-based incremental learning to continuous search spaces. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 418–427. Springer, Heidelberg (1998)
19. Yunpeng, C., Xiaomin, S., Hua, X., Peifa, J.: Cross entropy and adaptive variance scaling in continuous EDA. In: Thierens, D., et al. (eds.) Proc. of the Genetic and Evol. Comp. Conf. — GECCO–2007, pp. 609–616. ACM Press, New York (2007)

New Approaches to Coevolutionary Worst-Case Optimization

Jürgen Branke and Johanna Rosenbusch

Institute AIFB, University of Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de, johanna.rosenbusch@student.kit.edu

Abstract. Many real-world optimization problems involve uncertainty. In this paper, we consider the case of worst-case optimization, i.e., the user is interested in a solution's performance in the worst case only. If the number of possible scenarios is large, it is an optimization problem by itself to determine a solution's worst case performance. In this paper, we apply coevolutionary algorithms to co-evolve the worst case test cases along with the solution candidates. We propose a number of new variants of coevolutionary algorithms, and show that these techniques outperform previously proposed coevolutionary worst-case optimizers on some simple test problems.

1 Introduction

Many real life optimization problems involve some form of uncertainty, e.g., because they rely on forecasts, because they depend on an opponent's move, or because the solution eventually implemented is subject to manufacturing tolerances. In such cases, one typically searches for a *robust* solution. Often used criteria are a good expected quality or a low variance (see, e.g., (3, p. 127)). In the following, we consider the case that a user is interested only in a solution's worst-case performance, for example, because the application may include the risk of very severe consequences, such as death or bankruptcy.

Possible applications of worst-case optimization include engineering design (12), portfolio management (10) and scheduling (2, 6, 8)). There exist several ways to approach worst-case optimization, e.g., the calculation of reliability (4) or the use of an embedded EA to identify, for each solution, the worst-case (1). The latter implies great computational efforts which may render the approach infeasible in practice. An alternative and more efficient way to search for a robust solution *and* for its worst case simultaneously is provided by coevolutionary algorithms.

2 Coevolutionary Worst-Case Optimization

There exist several forms of coevolutionary algorithms (CEAs) but we consider only competitive, test-based CEAs which comprise one population consisting of solution candidates and one population forming the test cases (see, e.g., (7, 13, 14) for some early work).

CEAs offer several amenities. They do not need an objective, external metric to evaluate the solutions. Instead, individuals are evaluated by letting them interact with each other. Therefore, CEAs are applicable to problems where an objective criterion does not exist or cannot be computed. This is also the case in worst-case optimization, since the worst-case scenario is unknown, and a test with all possible scenarios may be impossible. CEAs are more efficient in that they use only a limited number of test cases to evaluate a solution. The population of test cases is furthermore selected *adaptively* because it coevolves with the solution candidates and therefore increases in difficulty as the solutions grow more powerful (5).

Over the course of research and application, CEAs have also shown some shortcomings. The first is a direct consequence of the lack of an objective metric: the real (objective) quality of a solution does not necessarily correspond to the subjective quality, i.e., the quality perceived by the algorithm. Furthermore, CEAs are susceptible to various pathologies such as evolutionary forgetting, cycling, disengagement, or overspecialization. For a detailed analysis of these pathologies as well as possible remedies, please refer to, e.g., (11).

CEAs have been applied to a wide range of problems but only few involve worst-case optimization. In (15) a so-called "nested minimax optimization" is performed. One population consists of various designs for a neural controller. The other population persists of plants, i.e., the scenarios in which the controller will be utilized. (2) uses a CEA to solve constrained optimization problems, written as min-max problems. One population evolves the parameter which is responsible for the minimization, the other population represents the parameter which maximizes. (6) and (8) apply worst-case CEAs to the area of scheduling, trying to find robust schedules. One population evolves the schedules, the other population evolves difficult problem instances or possible machine failures. Furthermore, (9) deals with the topic of worst-case optimization on a more theoretical level. We will discuss the basic idea of fitness assignment in these approaches in Section 4.

3 Coevolutionary Algorithm and Test Problems

The basic coevolutionary algorithm considered here uses two populations P_S and P_T. The solutions $s \in P_S$ attempt to minimize a function $F(s,t)$, while the test cases $t \in P_T$ are responsible for identifying the worst cases (i.e., attempt to maximize $F(s,t)$). Each individual is a single real number, and a standard $(\mu+\lambda)$ evolution strategy is used on both populations with mutation as the only variation operator. The mutation operator is additive Gaussian, and mutation probability is 100%.

We test the methods on two different functions. In each function, the solutions, denoted as s, aim at minimizing $F(s,t)$ while the test cases, t, aim at maximizing $F(s,t)$. The functions were designed in such way that the worst case is $t = s$ and the optimum solution is $s = 0$.

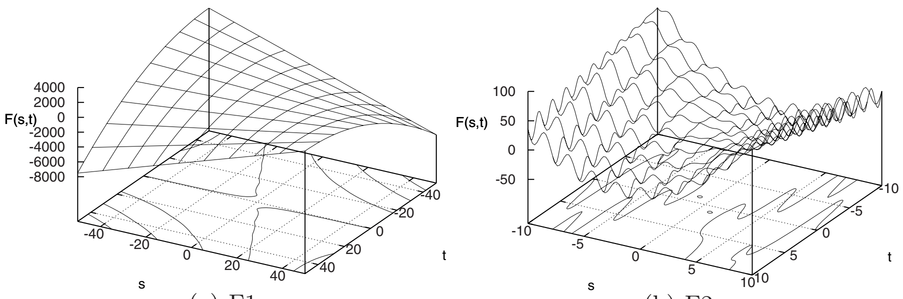
$$F1(s,t) = 2st - t^2, \qquad s \in [-50; 50], t \in [-50; 50] \tag{1}$$

The optimum of function $F1$ is stable because the solution candidates have no incentive to deviate from $(0, 0)$.

$$F2(s,t) = s^2 - s\cos(3(s - \frac{\pi}{2})) - (3|s-t| - (s-t)\cos(3((s-t) - \frac{\pi}{2}))),$$
$$s \in [-10; 10], t \in [-10; 10] \tag{2}$$

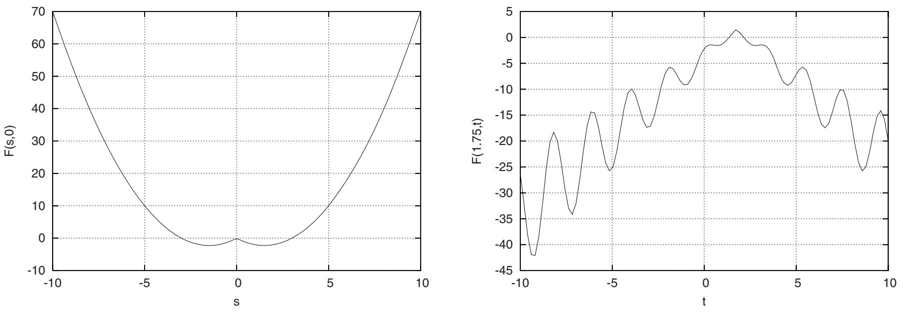
In contrast to function $F1$, $F2$ is much more rugged and since the point $(s = 0, t = 0)$ is not Nash, the optimum of $F2$ is not stable. Since the solution candidates aim to minimize, they have a very strong incentive to deviate as soon as the coevolutionary system comes close to $(0, 0)$. Because the test cases follow the solution candidates, both leave the optimum quickly (see Fig. 2).

For some visualizations of functions $F1$ and $F2$ see Figures 1 and 2.



(a) F1 (b) F2

Fig. 1. Visualization of test functions $F1$ and $F2$



(a) F2 for t = 0 (b) F2 for s = 1.75

Fig. 2. With Function $F2$, the solution candidates have an incentive to deviate from the optimum (left panel) which as a consequence, causes the test cases to follow (right panel)

4 Fitness Assignment Methods

In this section, we examine various fitness assignment methods, i.e., methods which calculate an individual's fitness based on the results of testing all solutions against all test cases, resulting in $|P_S| \times |P_T|$ function evaluations.

Let us denote the populations before selection as P_S and P_T, and the (smaller) populations after selection as P'_S and P'_T. Then, we distinguish between three different rankings/fitness values. By *real* ranking, we mean a ranking based on the solution's performance with respect to the true worst case (i.e., $t = s$). We call the ranking based on the populations before selection as *global* ranking, and the ranking based on the populations after selection the *local* ranking. W.l.o.g., best solutions have the lowest fitness, while for test cases, a higher fitness is assumed to be better. Note that the local fitness is always at least as good as the global fitness, as additional test cases can only worsen performance.

The solutions are always ranked according to their respective global worst case performance (over all test cases).

$$fit(s) = \max_{t \in P_T} F(s,t)$$

Furthermore, let us assume that solutions are numbered from 1 to n in order of increasing fitness (increasing global worst case values), i.e., the currently best perceived solution in the population is denoted by s_1.

In the following, we first describe two fitness assignment methods from the literature, namely the Maximin method and Jensen's method. Then, we continue to propose some new approaches.

Maximin Method. To rank the test cases, the classical approach is to also use the minimax principle, see, e.g., (2, 6, 8, 15). Since they represent the opposite perspective, the correct term is Maximin method.

$$fit_{Maximin}(t) = \min_{s \in P_S} F(s,t)$$

(8) shows, however, that this approach fails to find the optimum, if the concerned function is not a saddle-point function.

Jensen's Method. This approach is described in detail in (9). Jensen argues that the fitness of a test case should not only rely on the performance of that particular test case on P_S, but also on other test cases in P_T. If at least one solution exists, for which a test case forms a very difficult (or the worst) case, this test case should get a high fitness, even if it is easy to solve for the other solutions. Therefore in Jensen's approach, a test case's fitness equals the highest ranking it achieves if all test cases are ranked for each solution, the worst case having the highest rank. If a test case achieves this highest rank for $k > 1$ solutions, its fitness is additionally increased.

$$fit_{Jensen}(t) = \max_{s \in P_S} rank_s(t) + \frac{k}{|P_S| + 1},$$

where $rank_s(t)$ is the rank of solution s according to test case t.

The following methods are new, and select test cases one by one, in an iterative and greedy manner, trying to maximize the information about the individuals in P_S that the selected test cases provide. These test cases implicitly serve as a kind of memory, and influence the test cases future solutions will face.

Note that because we use a simple $(\mu + \lambda)$ selection strategy, the only relevant decision is which test cases survive to the next generation. However, the methods could be adapted to other selection methods in a straightforward way by assigning them ranks according to the order in which they are selected. Also, it is possible that the criterion to select the next test case is not unique. In this case, one of the test cases is added randomly.

Worst Case Method. The underlying idea of this method is that it is most important to keep the information about the worst cases of the best solutions (as these will be used to generate offspring). The method starts by going through all solutions $s_1 \dots s_n$ in order of increasing fitness, and, in each iteration, adding the corresponding worst case test case if it is not yet included.

Average Greedy Method. The Average Greedy method is based on the assumption that the local performance reflected by the selected test cases should be as close as possible to the global performance according to all test cases in the population. Therefore, the method starts by selecting the worst case of s_1. Then, it iteratively adds as next test case the one which maximizes the average local fitness of all solutions after adding the additional test case. More formally, if B denotes the set of test cases selected so far, it adds the test case $t' \in P_T$ which maximizes $\sum_{s \in P_S} max_{t \in \{B \cup t'\}} F(s,t)$.

Distance Greedy Method. The Distance Greedy method is based on the observation that a solution's fitness can only deteriorate if additional, more difficult test cases are found. To avoid that the best solution is no longer best in the subsequent iteration, it is attempted to maximize the difference in local fitness between the best and the closest competitors. Again, the method starts by selecting the worst case for s_1. Then, it iteratively adds the test case which, if added to the already selected solutions (B), maximizes the local fitness difference between the best solution and the second best (according to the ranking based on B). In case of ties, the difference between best and third best, fourth best, etc. is used as a criterion. Usually, this method leads to a correct local ranking of the best solutions, as the worst cases for these solutions are often selected first.

Ranking Greedy Method. Here, the motivation is to maintain the relative ordering of all solutions in P_S with only the selected solutions, i.e., to make the local ranking as consistent as possible with the global ranking. Again, the worst case of s_1 is always selected. Then, iteratively the test case is added to B that, if added, maximizes the correct number of relative orderings in the ranking specified by B.

Table 1 demonstrates how the different methods rank the test cases. The particular example consists of six solutions and six test cases, and assumes that

Table 1. Left: Evaluation of six solutions by six test cases (smaller numbers are better). Right: Fitness values assigned to test cases by the different methods (higher numbers are better).

	t_1	t_2	t_3	t_4	t_5	t_6
s_1	3	5	8	9	11	10
s_2	7	9	13	2	4	5
s_3	4	6	7	9	3	2
s_4	12	5	8	9	10	11
s_5	5	6	7	**8**	2	3
s_6	2	8	9	10	8	6

	t_1	t_2	t_3	t_4	t_5	t_6
Maximin	2	5	7	**2**	2	2
Jensen	5	4	5	**5.43**	5	4.29
Worst Case	3	1	2	**5**	4	0
Average Greedy	3	1	4	**5**	2	0
Distance Greedy	1	4	2	**5**	3	0
Ranking Greedy	2	1	4	**5**	3	0

in the case of ties, always the test case with the smaller index is chosen. As can be seen, the methods value the test cases quite differently, and it is not obvious which ranking is best. In any case, note that the Maximin method gives a very low evaluation to the test case causing worst case performance of the best solution (t_4, bold). Thus, it is likely that this test case does not survive to the next iteration, which, from our experience, would be very important. Jensen's method gives this test case the highest evaluation in this example, although this is not guaranteed in general. All our newly proposed methods include this test case with highest priority in the next population.

5 Metrics

Various metrics are used to analyze the performance of the different fitness assignment methods. The most important one is the *real fitness* of the generation's perceived best solution. Applying the knowledge that the real worst case is $t = s$, the real fitness can be calculated as $F(s_1, s_1)$.

The *correlation coefficient* between the real fitness ($F(s_1, s_1)$) and the subjective fitness of a solution ($fit(s_1)$) indicates the method's ability to keep real and subjective fitness linked to each other. It's measured across all individuals of the population.

A similar metric we designed is called *real quota*. It measures the fraction of the μ objectively (according to real fitness) best solutions the method succeeds to identify by counting how many of them are among the μ *subjectively* best solutions, i.e., whether the selection of μ parents is correct. A real quota of 1 means a perfect match.

Both metrics, correlation coefficient and real quota serve only for monitoring. The information they use is not accessible to the CEA and thus cannot be used to direct the fitness assignment process. A metric which uses only information that is acquirable for the CEA is the *global quota*. It counts how many of the μ subjective, i.e., global, best solutions are among the μ *local* best solutions. This metric gives information about the fitness assignment process in the test case population, which can be designed to optimize the global quota.

6 Empirical Results - Fitness Assignment

Experimental Setup. In this section, we use a $(10+20)$ evolutionary strategy with Gaussian mutation and step size 0.35. All values are mean values over 400 runs. Unless specified otherwise, the plotted values depict mean value and standard error of the respective metric.

Figure 3 displays the evolution of real fitness of the perceived best solution over time for test function $F1$. As can be seen, all fitness assignment methods except the method proposed by Jensen are able to converge to the optimum on this simple problem. The minimax method converges significantly slower, the newly proposed fitness assignments all perform similar. Additional experiments have shown that Jensen is also able to converge on this problem when allowed a larger population size.

The same plot, but for function $F2$, is shown in Figure 4. Here, the performance differences are much more significant. None of the algorithms is able to converge to the optimum, which was to be expected, since the function rewards deviations from the optimum. Average Greedy and Ranking Greedy perform best, followed by Distance Greedy and Jensen's method. The Worst Case method works very well in the first few iterations, but then suddenly deteriorates and converges to a level much worse than what had been obtained before. The good performance in the first phase can be explained by the uncompromising focus on the worst cases, driving the solution values down. The following ascent may be explained by an overspecialization in P_T. Very few test cases form the complete set of worst cases for all solutions, resulting in less than μ test cases with an assigned fitness. Therefore, some test cases are chosen randomly, substantially worsening the algorithm's performance.

The Maximin approach actually diverges and results in solutions worse than the random initial population.

Global Quota and Correlation Coefficient. The basic assumption behind the greedy approaches was that the performance of the CEA could be improved by maintaining, in the local information, as much of the global information as possible. The success of this idea is reflected in the sound performance of

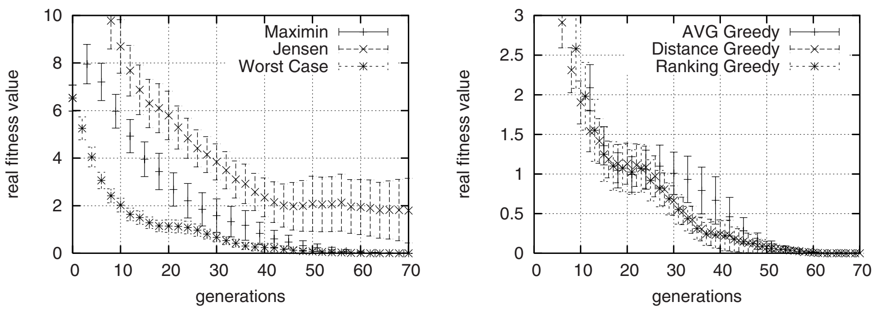


Fig. 3. Real fitness values and standard error for function $F1$ (optimum is 0.0)

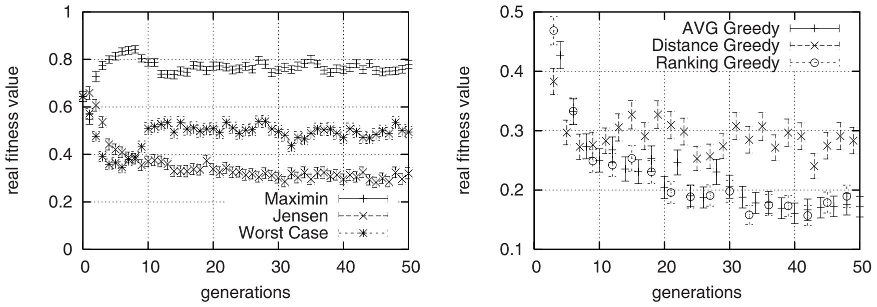


Fig. 4. Real fitness values and standard errors for function $F2$ (optimum is 0.0)

Table 2. Correlation coefficient and global quota ± standard error in generation 100 for function $F2$

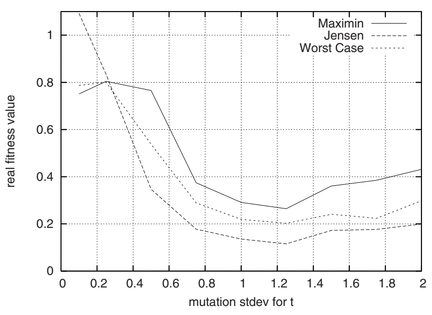
Method	Correlation Coefficient	Global Quota
Maximin	0.73 ± 0.017	0.54 ± 0.018
Jensen	0.8 ± 0.016	0.814 ± 0.007
Worst Case	0.71 ± 0.018	0.475 ± 0.018
Average Greedy	0.89 ± 0.01	0.87 ± 0.006
Distance Greedy	0.837 ± 0.013	0.84 ± 0.007
Ranking Greedy	0.9 ± 0.009	0.924 ± 0.006

the greedy methods on both problems. It can also be measured by the global quota and the correlation coefficient as reported in Table 2. The Ranking Greedy method was the last greedy method to be designed and it was especially developed to further improve the global quota, which was clearly successful. Nevertheless the Ranking Greedy method does not outperform the Average Greedy method regarding the real fitness, indicating that the connection between global quota, real quota and the correlation coefficient respectively seems to be more complex than expected. Table 2 shows that Average Greedy and Ranking Greedy have the same correlation coefficient, stating that both achieve about the same correlation between subjective and objective fitness.

7 Empirial Results - Mutation Step Size

The difficulty of function $F2$ lies in the fact that once the test case population converged to the worst case ($t = s$), the optimum is surrounded by a much more attractive area for the solutions. Therefore, they mainly circle around $(0, 0)$, always followed by the test cases. In order to drive the solutions back to $(0, 0)$, the test cases must "overtake" the solution value. This insight led us to test a mutation step size for the test cases larger than the mutation step size for the solutions.

To analyze the relation between the mutation step sizes of the two populations, the mutation step size of the solution population was fixed to a standard



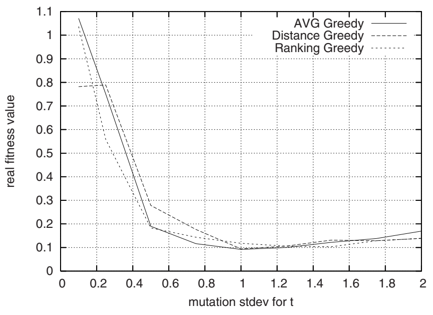


Fig. 5. Function $F2$: Best solution found in Generation 100. Mutation step size for P_S is fixed to 0.5 and varies for P_T.

deviation $\sigma = 0.5$ while various values were tested for the test case population. The real worst case fitness of the last generation's perceived best solution was plotted for each combination. The mean values over 400 runs can be seen in Fig. 5. The standard errors are very small, and have been omitted in the plot for clarity. Best performance is reached if the test cases' mutation step size is a bit more than double of the solutions' step size. So, our initial assumption has been confirmed. Increasing also the solution population step size to the higher value again lead to worse performance (not shown).

8 Conclusion

Coevolutionary algorithms seem an efficient and promising approach to worst-case optimization. In this paper, we have proposed and analyzed a number of variants of coevolutionary algorithms. The focus of our study was on new ways to determine the fitness of the test cases. Here, we proposed several greedy mechanisms which aim at preserving as much worst-case information about the good solutions as possible after selection. As has been shown empirically, the new methods significantly outperform previously proposed fitness assignment schemes on the suggested test functions.

Besides, we have experimented with different mutation rates for the solution and test case populations, and found that it is beneficial to choose a higher mutation rate for the test population than for the solution population.

Overall, this paper has proposed several novel and promising ways to improve the performance of coevolutionary worst-case optimizers. As a next step, the obtained results should be confirmed on a variety of additional test problems. Also, it would be straightforward to use the various methods in a lexicographic order, and switch from one to another in case of ties.

Acknowledgement. We would like to thank Elena Popovici from Icosystem Inc. for interesting discussions.

References

1. Avigad, G., Branke, J.: Worst-case robustness and related decision support. In: Genetic and Evolutionary Computation Conference, ACM Press, New York (to appear)
2. Barbosa, H.J.C.: A coevolutionary genetic algorithm for constrained optimization. In: Congress on Evolutionary Computation, vol. 3, pp. 1605–1611 (1999)
3. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell (2001)
4. Daum, D.A., Deb, K., Branke, J.: Reliability-based optimization for multiple constraints with evolutionary algorithms. In: Congress on Evolutionary Computation, pp. 911–918. IEEE Computer Society Press, Los Alamitos (2007)
5. de Jong, E.: The maxsolve algorithm for coevolution. In: Conference on Genetic and Evolutionary Computation, pp. 483–489. ACM Press, New York (2005)
6. Herrmann, J.W.: A genetic algorithm for minimax optimization problems. In: Congress on Evolutionary Computation, vol. 2, pp. 1099–1103. IEEE Computer Society Press, Los Alamitos (1999)
7. Hillis, D.W.: Co-evolving parasites improve simulated evolution in an optimization procedure. Physica D 42, 228–234 (1990)
8. Jensen, M.T.: Finding worst-case flexible schedules using coevolution. In: Spector, L., et al. (eds.) Genetic and Evolutionary Computation Conference, pp. 1144–1151. Morgan Kaufmann, San Francisco (2001)
9. Jensen, M.T.: A new look at solving minimax problems with coevolutionary genetic algorithms. Applied Optimization 86, 369–384 (2004)
10. Korn, R., Steffensen, M.: On worst-case portfolio optimization. SIAM Journal on Control and Optimization 46(6), 2013–2030 (2007)
11. Luke, S., Wiegand, R.P.: When coevolutionary algorithms exhibit evolutionary dynamics. In: Barry, A.M. (ed.) GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, pp. 236–241. AAAI Press, Menlo Park (2002)
12. Ong, Y.-S., Nair, P.B., Lum, K.Y.: Max-min surrogate-assisted evolutionary algorithm for robust design. IEEE Transactions on Evolutionary Computation 10(4), 392–404 (2006)
13. Pagie, L., Hogeweg, P.: Information integration and red queen dynamics in coevolutionary optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 2, pp. 1260–1267 (2000)
14. Paredis, J.: Coevolutionary computation. Artificial Life 2(4), 355–375 (1995)
15. Sebald, A.V., Schlenzig, J.: Minimax design of neural net controllers for highly uncertain plants. IEEE Transactions on Neural Networks 5(1), 73–82 (1994)

Bio-inspired Search and Distributed Memory Formation on Power-Law Networks

Tathagata Das, Subrata Nandi, Andreas Deutsch, and Niloy Ganguly

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal - 721302, India

Abstract. In this paper, we report a novel and efficient algorithm for searching P2P networks having a power law topology. Inspired by the natural immune system, it is a completely decentralized algorithm where each peer searches by sending out random walkers to a limited number of neighbors. As it finds other peers having similar content, it restructures its own neighborhood with the objective of bringing them closer. This restructuring leads to clustering of nodes with similar content, thus forming P2P *communities*. Alongside, the search algorithm also adapts its walk strategy in order to take advantage of the community thus formed. This search strategy is more than twice as efficient as pure random walk on the same network.

1 Introduction

Due to the dynamic nature of large scale peer-to-peer(P2P) networks, the search algorithms used for such system need to be decentralized, self-adjusting and robust against rapidly changing system environments. Borrowing ideas from the living nature has long been a fruitful research theme in various fields of communication engineering. The inspiration of design patterns from biological systems has been well exploited in our work [4], where we have introduced practically relevant algorithms for distributed computing that naturally inherit the desirable properties of biological systems including adaptivity and robustness. In this paper, we have taken inspiration from different properties of the humoral and secondary immune system to design and test different random walk and proliferation based search and community formation algorithms in power law P2P networks.

The idea of forming P2P communities to improve search efficiency is an ongoing research field. Many groups have explored this concept with very specific types of networks (distributed libraries for example [6]) while other have applied it to Erdos-Renyi networks [7]. However, except some of our previous works [1], there has hardly been work on algorithms where the search process itself triggers community formation. The idea is that the network as a whole gets trained/acquires memory as search progresses.

Our previous work [1] was based on a grid-based topology. The algorithms developed there could not be ported in a more realistic power law network[8].

Here we present completely new algorithms that have been developed based upon a much more thorough understanding of the effect of various dynamics performed on the network. We first apply different varieties of random and greedy search mechanisms in order to understand the dynamics. Finally, we suggest an algorithm which consists of a healthy mix of random and greedy walking. We show that it performs far better then conventional random walk schemes.

Section 2 discusses in detail the behavior of the immune system and our inspirations from it. Section 3 describes the model of the P2P network that we use and the details of the various algorithms. Their performance in simulations is analyzed in section 4 and a final algorithm based on these results is proposed in section 5. Finally, we conclude in section 6 with the possible ways of improving these results.

2 Biological Inspiration: The Immune System

The immune system displays a number of amazing behaviors and attributes that can be an inspiration in providing robust solutions to a number of well known technological problems. The behavior can be distinctly attributed to two different parts of the immune system - the humoral (innate) immune system and the secondary immune system. Each of them has been a source of inspiration as described in the following section.

2.1 Humoral Immune System

In our earlier works [1],[2],[4] we had proposed a search algorithm for peer-to-peer networks that is inspired by the simple and well known mechanism of the humoral immune system where B cells upon stimulation by a foreign agent (antigen) undergo proliferation generating antibodies. Proliferation helps in increasing the number of antibodies while mutation implies a variety of generated antibodies. Consequently the antibodies can efficiently track down the antigens (foreign bodies). This is modeled by considering the query message packet as an antibody which is generated by the node initiating a search whereas antigens are the searched items hosted by other constituent members (nodes) of the P2P network. Similar to the natural immune system, the packets walk through the network followed by proliferation based upon the affinity measure between the message packets and the contents of the node visited.

In our current work, we have analyzed the dynamics of the packet movement in greater detail, in order to determine the parameters that control the degree of movement. The movement of the antibodies can either be a purely random walk or it maybe a biased random walk similar to the *adhesion*-based movements of cells within the *extracellular matrix* (ECM) [5]. This phenomenon of cell movement guided by adhesion is called *Haptotaxis*, and the movement of the query in a P2P network resembles a haptotactic cell movement in the ECM [4]. Based on this phenomenon, we model the two basic types of movement strategies as *random* (unbiased movement) or *greedy* (movement biased by properties of the neighborhood), respectively. The query movement itself can be subdivided into two distinct phases - *general movement / walk* and *proliferation*, each of which can be independently performed either randomly or greedily. The details of the four algorithms inspired from such random and greedy approaches will be discussed in section 3.

2.2 Secondary Immune System

We have also taken inspiration from the secondary immune response mechanism, which has the capability to develop memory over time and accordingly the antibodies produce a quicker response [3]. This decentralized memory is modeled in our P2P network by restructuring the connections in the network in order to form virtual communities of nodes having similar items (antigens). Each search initiates a rewiring of the network towards community formation based upon the information content of the participating nodes. Due to this community formation, the network gets trained with time to find similar nodes with greater efficiency. In essence, the entire network acts as a large memory which is able to optimize the search process. The exact details of the community formation process are dealt with in section 3.

Based on these inspirations we were motivated to test out the concepts and understand the underlying dynamics in order to decide what level of randomness or greediness is optimal for best performance in P2P networks.

3 Model and Algorithms

In this section we first define the model of the P2P network that we will use in the following, then we discuss the detailed implementation of the search and community formation algorithms.

3.1 Peer-to-Peer Model

We assume a realistic power-law topology for the P2P network (as most of the existing P2P networks exhibit a similar topology). Also, in order to form content-based communities, we have classified the information content of the peers into abstract subcategories. The details are provided below.

Topology and Network Load. According to the characteristic heavy tailed nature of power law networks, few nodes have high degrees while the majority of the nodes have low degrees. These initial connections are assumed to form a connectivity layer among the nodes and are hence termed as *Connectivity Edges*. New edges that are added to the network with the intention of forming community structures over the connectivity layer are called *Community Edges*.

For the purpose of our analysis, we consider the degree of a node as a measure of its continuous bandwidth usage, assuming that a low bandwidth consuming gossip protocol maintains the communication between the neighbors. Hence, there is a limit to the total number of edges it can have. In other words, each node can sustain only a limited number of new community edges. This increase in network load is measured relative to the initial network degree (that is, the degree corresponding to its connectivity edges). This measure is termed as X where $X = \frac{New\ Degree\ -\ Initial\ Degree}{Initial\ Degree}$. The maximum network load that each node can tolerate is assumed to be X_{max} times the initial network load (that is, the initial degree). Also, during the search protocol, there will be bursts of high bandwidth usage when a node needs to communicate with its neighbors. This is also limited by a maximum number of neighbors that a node can contact in a single burst of communication. Let this limit be known as Y_{max}.

Profile Distribution. In a file sharing P2P network, each node shares some data with other nodes in the network. These data are categorized into abstract categories called *Information Profiles*. The profiles (P_I) therefore reflect the informational content as well as the informational interest of the user. A profile is represented in our system as a m-bit binary value, thus producing 2^m distinct categories. These profiles are distributed among the nodes following Zipf's law[8] with the idea that some categories of data are highly popular whereas others are not.

Search and Matching. A search query is defined as a m-bit binary value, which is taken to be equal to the information profile P_I of the node that is initiating the search. This is based on the simple idea that the user of the node would like to search for items that fall into the same category as his own information content. In order to find nodes having similar content, the query packet is forwarded in the network according to the rules set by the search algorithm. Each node that encounters the query packet tries to match its own profile with the queried profile. When a node is found whose information profile exactly matches the query profile, it is said to be a *search hit* and the initiator node and matched node are said to be *similar* nodes.

3.2 Algorithms

As indicated in section 2, we would like to test out the four major types of the proliferation-based search algorithms – named $\mathcal{RR}$, $\mathcal{RG}$, $\mathcal{GR}$ and $\mathcal{GG}$. In this section, we describe these algorithms in full detail. As mentioned earlier, there are two distinct processes in the algorithms – Search and Community Formation.

Search — Any node in the networks can start a search query. Let us say, it is initiated at a node U. It sends a search query message M to a few of its randomly selected (at most Y_{max}) neighbors, carrying the information profile (P_I) of U as the query profile to be searched. This message packet walks through the network until it comes across a node whose information profile matches with the queried profile. Then it is said to have made a *search hit*. Let that node be called node A. Following the search hit, A performs two operations - *Proliferation* and *Community Formation*. A proliferates (replicates) the query to a number of its neighbors (at most Y_{max} neighbors) with the aim of making a more intensified search in its vicinity. This is done to exploit the fact that due to community formation, nodes similar to A (hence similar to U) should be present in the neighborhood of A. Moreover, the general walk is further optimized by making each query packet store the nodes it has traversed through, so that they are avoided while forwarding the packets.

Table 1. Neighbor selection strategies in different search algorithms

	Search algorithms			
Neighbor selection strategy	$\mathcal{RR}$	$\mathcal{GR}$	$\mathcal{RG}$	$\mathcal{GG}$
During query forwarding	Random	Greedy	Random	Greedy
During proliferation	Random	Random	Greedy	Greedy

The neighbor selection process for general walking and proliferation decides the randomness / greediness of the overall walk mechanism. The neighbors for the general query forwarding as well as during proliferation can be selected in two ways: **Random -** neighbors are chosen randomly without any bias i.e. without considering the type of edge through which it is connected; **Greedy -** neighbors connected by community edges are preferred for selection over other neighbors. In case of query forwarding, only one neighbor is selected in this manner, whereas multiple neighbors are selected in case of proliferation.

Various permutations of these general walk and proliferation schemes lead to four different types of searches. As shown in table 1, they have been named by two letters based on the $\mathcal{R}$andom or $\mathcal{G}$reedy scheme used. The first letter represents the scheme used for general walk and second letter for the proliferation scheme. We next explain the latter process, that is, Community Formation.

Community Formation — Whenever there is a search hit, we want to evolve the topology in order to increase the probability of the next query reaching the node A from U. This can be ensured simply by connecting the similar nodes U and A with a new community edge. This brings the similar nodes within one hop distance of each other, thus increasing the probability of reaching it in the next search attempt. On the other hand, due to the network load limit of X_{max}, the algorithm is forced to delete edges when a new edge AB causes the network load of A and/or B to exceed its limit. Hence, we delete the edge with the following strategy. If both A and B exceed limits because of the new edge AB, then this edge is removed. If either A or B exceeds the limit, then another community edge is randomly selected for deletion from the corresponding node. Furthermore, each edge is added with a probability of $Prob_{add}$. This regulates the speed of addition and prevents the network load of each node from reaching its limit very fast. Hence each node gets 'time' to learn and the network does not unnecessarily undergo a huge amount of churn to stabilize. It must also be noticed that we are churning only the community edges, and not the connectivity edges, which ensures that the whole network remains connected at all times.

3.3 Evaluation Criteria

We will like to evaluate the performance of these algorithms based on the following criteria.

Search related metrics. Let us assume that the i^{th} search produces a total of h_i search hits using a total of p_i packets. Let the total number of nodes similar to the initiator node (that is the maximum possible search hits) be H_i. Let the search be performed n times. The search-related metrics are defined as follows:

Total Hit Count: Average number of hits (similar nodes) found in each search, i.e. $\frac{1}{n}\sum_i h_i$

Efficiency: Average number of hits per search packet, i.e. $\frac{1}{n}\sum_i \frac{h_i}{p_i}$

Similar Node Coverage: Average fraction of all the similar nodes present in the network that is returned in each search, i.e. $\frac{1}{n}\sum_i \frac{h_i}{H_i} \times 100$.

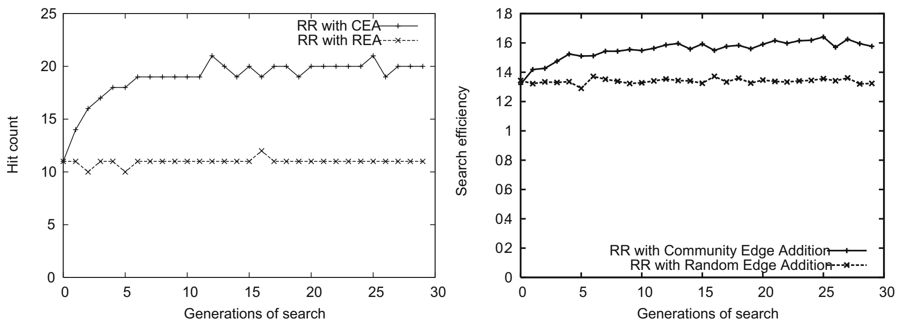
Metrics related to community formation. The community edges make connections between similar nodes only. If we consider nodes of a particular profile, then these edges form a community overlay network over these nodes. The size of the largest connected component (LCC) in a network is generally considered as a measure of its connectedness. Since, we desire that all the nodes of a profile are well connected by the community overlay network, we take the LCC of the network as a measure of the '*goodness*' of the community structure. It is expressed in terms of the percentage of nodes of each particular profile that lie within the LCC. This is averaged over all the profiles in the system, and is termed as *Average LCC* of the community structure.

4 Simulation and Results

In order to test out the performance of the proposed algorithms, we resorted to simulations whose details are as follows.

4.1 Simulation Scheme

For simulating our algorithm, we took a power-law network of 1000 nodes, generated using the Barabasi-Albert preferential attachment method, which gave us



(a) Total number of search hits (b) Search efficiency

Fig. 1. Performance of $\mathcal{RR}$ search using Community Edge Addition (CEA) compared to Random Edge Addition (REA)

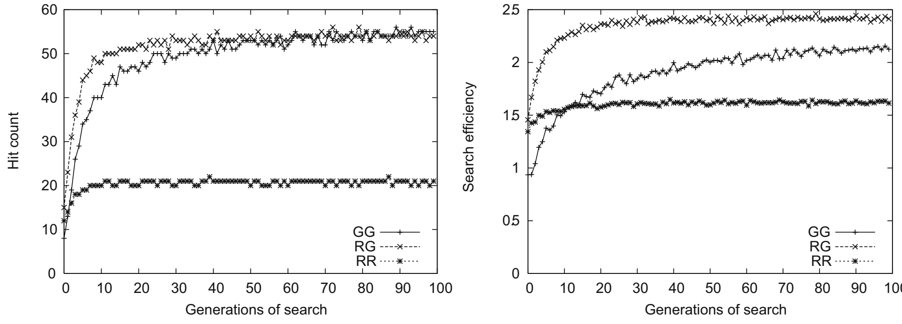
a gamma of approx 2.0. 16 profiles ($m = 4$) were distributed among the nodes by Zipf's law with a gamma of 0.8. Each search query is propagated in this network up to 15 hops. A set of search queries (generally 200) executed on random nodes constitute a generation and all performance metrics were averaged over a generation. Edge addition probability $Prob_{add}$ is 0.3, while the network load limit X_{max} is 1.5. Y_{max} was chosen to be 3 nodes. A number of generations performed on the same network constitute a simulation. Multiple simulations are performed on different profile distributions for averaging the performance of the algorithm.

In order to prove the importance of community formation, we performed a fairness test by comparing the performance of network formed through community edge addition (CEA) with an equivalent network. In this equivalent network, we start from the same initial power law network as the actual simulated network, and we compensate for the increase in the edge count of the latter (due to community edge addition) by randomly adding an equal number of edges (that is, random edge addition (REA)) in the equivalent network.

4.2 Results and Analysis

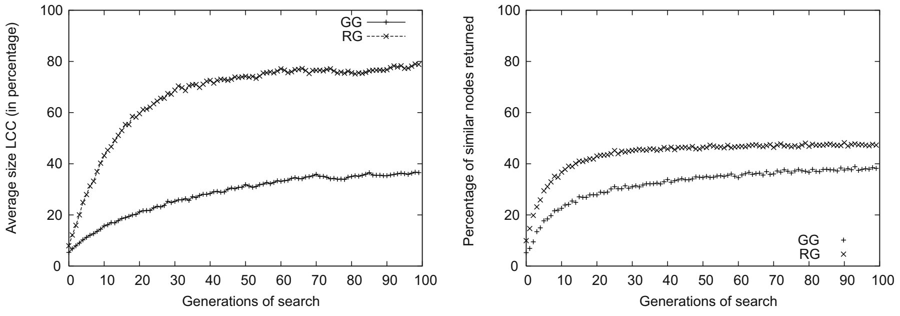
First of all, we present the performance of $\mathcal{RR}$ with community edge addition versus random edge addition on an equivalent graph. Figure 1(a) shows that as generations of search progress, the total number of hits returned by community edge addition increases steeply compared to random edge addition, finally producing an average of 20 hits compared to 11 by the latter. In terms of efficiency, the former performs up to 20% better than the latter (Fig. 1(b)). This clearly proves that strategic addition of edges by community formation improves the search efficiency, unlike random addition edges.

Next we present the performance of $\mathcal{RG}$ and $\mathcal{GG}$ (we omit the result of $\mathcal{GR}$ due to lack of interesting inferences). All these cases undergo community edge addition. Figure 2(a) shows that on average, the number of results brought by both types of greedy-proliferation based searches are comparable, while being more than 2.6 times better than that of $\mathcal{RR}$. In terms of search efficiency, $\mathcal{GG}$ and $\mathcal{RG}$



(a) Hit count (b) Search efficiency

Fig. 2. Performance of $\mathcal{RR}$, $\mathcal{RG}$ and $\mathcal{GG}$ wrt hit count and search efficiency



(a) Average LCC (b) % of similar nodes returned

Fig. 3. Correlation between LCC size & % of similar nodes returned in $\mathcal{RG}$ and $\mathcal{GG}$

perform about 30% and 50% better than $\mathcal{RR}$, respectively. Also, $\mathcal{GG}$ saturates much slower compared to $\mathcal{RG}$. Both these figures confirm without doubt the importance of greedy walking in proliferation. This is actually obvious – only by greedily choosing the community edges can the already formed community be efficiently searched.

The most obvious question that arises is - what produces the difference in the search efficiencies of $\mathcal{GG}$ and $\mathcal{RG}$? This is primarily because of the extent of community formation in both cases. To quantitatively measure the community formed between nodes of a particular profile, we calculate the size of the largest connected component (LCC) in terms of the fraction of the similar nodes it contains. The larger this fraction, the more well connected they are. Referring to Fig. 3(a), we see that the LCC in case of $\mathcal{RG}$ encompasses around 80% of all the similar nodes while it is just 40% in case of $\mathcal{GG}$. Greedy general walking in $\mathcal{GG}$ is unable to produce as good a community structure as the random walking in $\mathcal{RG}$, since it directs all the query packets into already discovered areas of the network and hence inhibiting the exploration (that is, node discovery). But on the other hand, $\mathcal{RG}$ is also not able to exploit the good community structure created, as it is returning a smaller fraction of similar nodes compared to that present in the LCC. Refer Fig. 3(b), $\mathcal{GG}$ is finding almost all (95%) the nodes that constitute the LCC (36%), while $\mathcal{RG}$ returns just around 60% of all such nodes in LCC (79%). To summarize, while a random general walk has a better performance in terms of node discovery and node retrieval, greedy general walk is better at efficiently searching the already discovered nodes. Hence, it will be beneficial if we are able to develop a search algorithm that embraces the best of both.

5 An Approach to Self-Adjusting Search ($\mathcal{SA}$)

Extending the idea of antigens and antibodies further, we want to design an algorithm that has the intelligence to adjust itself between two phases - Exploratory Phase and Search Phase [9]. In the former phase, the antibodies would explore the entire network in order to find the location of antigens. In the latter phase, when the antigens have been located, it would like to redirect all its effort towards the affected areas. In terms of our problem, our search algorithm should, in the initial stages, explore the graph with maximum probability (for developing the best community structure as soon as possible) and in the later stages search the network with maximum efficiency. In other words, it must be able to identify automatically whether it should put the maximum effort in exploring the network or in searching the network efficiently. We propose such an algorithm in the next section.

5.1 Algorithm

As evident in earlier results, $\mathcal{RG}$ performs a better exploration of the network, while $\mathcal{GG}$ performs a better search of the already explored regions. Each of the algorithms is individually suited for each of the two phases, respectively. So we need to design an algorithm that can adjust itself based on the phase of the system, in a decentralized manner. The key requirement for designing such an algorithm is to identify a property / parameter in the network based on which we can control the randomness / greediness of the search process.

In order to make the search tunable to random or greedy schemes, each query packet now holds another parameter - *Random Walk Probability* (P). At the time of initiation of the search, the value of the probability is set by the initiator node. This probability is also copied to the new packets created at the time of proliferation. Based on this probability, the non-matching nodes, through which the packets pass, will either forward the packet randomly (like $\mathcal{R}*$) or greedily ($\mathcal{G}*$). In the matching nodes, the behavior is always the same - greedy proliferation (as in $*G$). The probability can be set to different values between 0.0 and 1.0 to get a behavior in between pure $\mathcal{RG}$ and pure $\mathcal{GG}$.

Next, we need to choose a suitable parameter for determining the phase of the system in a decentralized manner. We have chosen this to be the X value of the node. If X is low, then it means that the node has the capacity of accepting new community edges and expanding the community structure. In that case, it should try to explore the network for previously undiscovered similar nodes with a higher probability. Conversely, when X is high and near its limiting value, its capacity of adding to the community structure is low. Therefore, instead of exploring, it should try to efficiently search the community structure that has already been formed around it. More formally, the probability of random walk is calculated as $Prob_{random} = 1 - \frac{X_A}{X_{max}}$ where $X_A = X$ of the node A that is initiating the search. The overall behavior would be as we desire - initially, when X is 0 for all the nodes, it will behave like pure $\mathcal{RG}$. Later as the X of all the nodes reach X_{max}, the probability of random walk reduces to zero, that is, it performs pure $\mathcal{GG}$ on an optimal community structure.

5.2 Simulation Results

Figures 4(a) and 4(b) reflect the superiority of $\mathcal{SA}$ scheme. The scheme is able to produce the best possible community structure as fast as $\mathcal{RG}$. Side by side, it overcomes the shortcomings of $\mathcal{RG}$ by being able to find almost all the similar

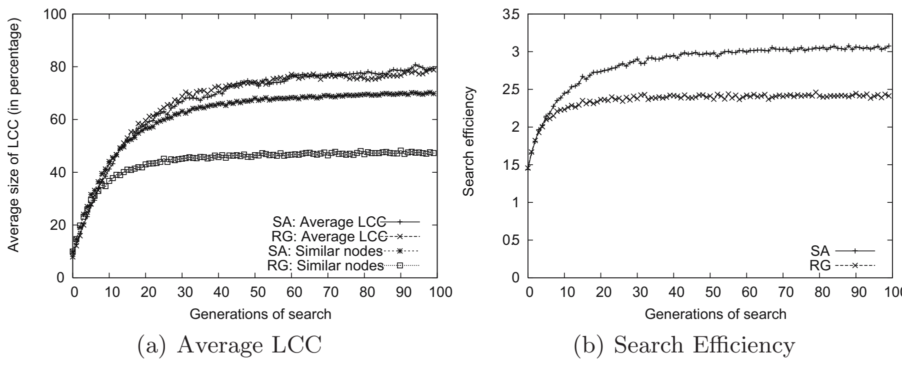


Fig. 4. Performance of Self-Adjusting Search

nodes in the LCC. Refer Fig. 4(a), $\mathcal{SA}$ is finding around 90% of the similar nodes that constitute the LCC, while $\mathcal{RG}$ returns just around 60% of all such nodes, thus producing almost 50% improvement. Finally, we find that the search efficiency of $\mathcal{SA}$ is about 30% better than $\mathcal{RG}$ (and more than 130% better than $\mathcal{RR}$ with REA).

6 Conclusion and Future Work

This paper has presented a community-based search algorithm applicable on power-law network which derives its inspiration from natural immune systems. Detailed study of the dynamics of the walk has been done which resulted in an elegant time-varying algorithm. The final algorithm, like the immune system, consists of exploration and search (healing) phase. The algorithm outperforms by far any conventional system and may have far reaching impact in designing efficient P2P communities in the future. A rigorous testing and fine tuning of the algorithm will be the main focus of our future work.

References

1. Ganguly, N., Canright, G., Deutsch, A.: Design Of An Efficient Search Algorithm For P2P Networks Using Concepts From Natural Immune Systems. In: Int'l Conf. on Parallel Problem Solving from Nature (2004)
2. Ganguly, N., Brusch, L., Deutsch, A.: Design and analysis of a bio-inspired search algorithm for peer to peer networks. SELF-STAR: Self-* Properties in Complex Information Systems (2005)
3. Klarreich, E.: Inspired by immunity. Nature 415, 468–470 (2002)
4. Babaoglu, O., Canright, G., Deutsch, A., Caro, G.D., Ducatelle, F., Gambardella, L., Ganguly, N., Jelasity, M., Montemanni, R., Montresor, A.: Design Patterns from Biology for Distributed Computing. ACM Transaction of Autonomous and Adaptive Systems 1(1) (September 2006)

5. Dickinson, R.B., Tranquillo, R.T.: A Stochastic Model for Adhesion-mediated Cell Random Motility and Haptotaxis. J. Math. Biol. 31(6), 563–600 (1993)
6. Asvanund, A., Krishnan, R.: Content-Based Community Formation in Hybrid Peer-to-Peer Networks. In: SIGIR Workshop on P2P Information Retrieval (2004)
7. Khambatti, M., Ryu, K., Dasgupta, P.: Structuring P2P Networks Using Interest-Based Communities, Databases, Information Systems & P2P Computing (2003)
8. Newman, M.: The structure and function of complex networks. SIAM Review (2003)
9. Murray, J.D.: Mathematical Biology. Springer, Heidelberg (1989)

Enhancing the Efficiency of the ECGA

Thyago S.P.C. Duque, David E. Goldberg, and Kumara Sastry

Illinois Genetic Algorithms Laboratory,
University of Illinois at Urbana Champaign,
104 S. Mathews Ave, 117 Transportation Bldg,
Urbana, IL, USA
{thyago,deg}@illigal.ge.uiuc.edu,
kumara@kumarasastry.com
http://www.illigal.uiuc.edu/web/

Abstract. In this paper we show preliminary results of two efficiency enhancements proposed for Extended Compact Genetic Algorithm. First, a model building enhancement was used to reduce the complexity of the process from $O(n^3)$ to $O(n^2)$, speeding up the algorithm by 1000 times on a 4096 bits problem. Then, a local-search hybridization was used to reduce the population size by at least 32 times, reducing the memory and running time required by the algorithm. These results are the first steps toward a competent and efficient Genetic Algorithm.

Keywords: Estimation of Distribution Algorithms, ECGA, Model Building, Efficiency Enhancement.

1 Introduction

Evolutionary Algorithms (EA) [1] [2] have been successfully used in several different applications involving search, optimization and machine learning problems. Goldberg [3] presents a design-decomposition methodology for successfully designing scalable Genetic Algorithms (GAs). A GA that can solve hard problems accurately, efficiently and reliably is called a competent GA. These GAs can solve problems that are intractable for other algorithms in a tractable polynomial time.

However, to solve large scale problems it is oftentimes necessary to enhance the efficiency [4] of the algorithm. Some common approaches include parallelization [5], hybridization [6] [7] [8], time continuation [9], and evaluation relaxation [10].

The recent success of the Compact Genetic Algorithm (cGA) [11] on solving a billion bit noisy optimization problem [12] [13] has proven that GAs, if properly designed and optimized, can solve difficult large problems.

The objective of this work is to reproduce this success, creating an efficient and competent EA, capable of solving large scale difficult problems. Particularly, we are interested in creating an algorithm that can deal efficiently with problem sub-structures, solving a broader class of problems than the class the cGA can handle. This paper presents a proof of principle on the importance of efficiency enhancements for GAs. We present the first steps toward such efficient and competent EA and discuss future directions for the next steps.

2 The Extended Compact Genetic Algorithm

The Extended Compact Genetic Algorithm (ECGA) [14] is an evolutionary algorithm in the class of Estimation of Distribution Algorithms (EDAs) [15]. These algorithms substitute the selection-crossover-mutation process, common to GAs, by a selection-model-building-sampling process. Some examples of EDAs include the cGA [11], that uses probability vector as a model, and the Bayesian Optimization Algorithm (BOA) [16] that uses a Bayesian Network as a model, being able to represent which variables are linked together.

The ECGA uses the Marginal Product Model (MPM), which can be divided into two components, (I) a partition over the variables, defining which variables are independent and which variables are linked, and (II) a probability distribution over each partition.

The ECGA is based on the principle that the estimation of a good model for the population is equivalent to the linkage learning [2] [14] process. It searches the space of possible partitions to find an appropriate one and tunes the probability distribution to match the data. The ECGA uses the minimum description length (MDL) principle as learning bias, which means that the cost of representing the whole population under the compression induced by the model (Compressed Population Complexity - CPC), together with the cost of representing the model itself (Model Complexity - MC) should be minimal.

The ECGA chooses a partition that appropriately models the sub-problem structure by greedily optimizing the Combined Complexity Criterion (CCC) using Algorithm 1, which assumes that each of the variables are independent and them evaluates all possible pair wise merges and pick the best until no merging can improve the CCC. After the partition is determined, the probability distribution is estimated simply by counting the frequencies in the population. For detailed information about the ECGA and the CCC, please refer to [14].

The ECGA main loop can be summarized by the following steps. First, it generates a random population and then repeats the process of evaluation, selection, model-building, and sampling until any convergence criteria is satisfied.

Algorithm 1. Greedy search for an appropriated model in the ECGA

```
1. Start assigning each variable to an independent partition
2. Repeat:
3.   For each pair of partitions:
4.     Merge that pair
5.     Evaluate the CCC of the model
6.     Undo the merging
7.   Merge the pair that induced the smaller CCC if any
8.   End the search if no improvement is possible
```

3 Efficiency Enhancements for the ECGA

The methodology proposed by Goldberg [3] allows us to design competent GAs, which can solve hard problems in polynomial time. The same methodology can be used to design efficiency enhancements for GAs [4], which can be divided in four main categories: Parallelization, hybridization, time continuation and evaluation relaxation.

In a very superficial view, parallelization deals with the division of the GA in several processors. Hybridization deals with the integration of GAs with other search procedures. Time continuation deals with the tradeoff among using a larger population for short time or smaller population for long time and evaluation relaxation deals with the tradeoff among having a noisy and cheap evaluation against an accurate and expensive one.

This paper proposes two extensions to improve the performance of the ECGA. One of them follows the cited methodology; it is the hybridization of the ECGA with a local search procedure that will work as a preprocessing mechanism. The other extension addresses the model building process, which according to running time profiling information consumes more than 90% of the computational time. The profiling information was generated using the GNU gprof utility for UNIX systems.

All results presented in this section use the mk-trap problem [9] with trap size (k) of 4 as benchmark problem. The mk-trap is an additively separable problem with m sub-problems, each of them being a trap function of k bits. The tournament size was fixed to 16 for all experiments. The population size and number of sub-problems (m) varies on the experiments.

3.1 Improving the Model Building Process

The ECGA is a competent genetic algorithm, but it is computationally expensive. The first step to improve its performance was to profile the code and to determine which are the most time consuming steps of the algorithm. The result of such profiling showed that the model building process took more than 90% of the computational time. This concerning aspect lead us into a search for model building alternatives to reduce this time.

The first possible approach to that problem would be the use of a cache structure, already used on [17], which sacrifices memory to get runtime improvements. However, this improvement is not sufficient for large problems and memory may also become a limiting resource.

Reviewing Algorithm 1, we point that line 2 introduces a loop that depends on the size of the chromosome (l). Line 3 introduces iteration over pairs of variables in the string, which can be re-written as: “For each variable; For each other variable”, introducing a order two iteration over l. The overall complexity is O(l^3).

One possible way to reduce the runtime would be to finish the loop in line 3 before it evaluate all pairs. Instead of searching for the best merging among all pairs, it might be enough to accept any merge that improves the CCC and stop the loop. This approach works for tight coded problems [2], but fails for loose coded ones when the population size is close to the minimum necessary for the ECGA, since spurious relations among actually independent variables are expected show up by just chance. Using a larger population is a way to bypass this problem but this price overcomes the benefits.

A successful approach would have to reduce the overall complexity of the algorithm. This is achieved using the Algorithm 2. Fundamentally, instead of evaluating the CCC over all pairs of partitions, an O(l^2) step, Algorithm 2 selects one partition (a) and evaluates the CCC only over the pairs that include this partition, reducing the complexity of this step to O(l).

Algorithm 2. Improved greedy model building for the ECGA

```
 1. Start assigning each variable to an independent partition
 2. Repeat:
 3.   Choose a partition (a)
 4.     For each other partition (b):
 5.       Merge a and b
 6.       Evaluate the CCC of the model
 7.       Undo the merging
 8.    Merge the pair that induced the smaller CCC if any
 9.    With some small probability pb, break any partition
10.    Cool down the breaking probability
11.    End the search if no improvement is possible
```

As mentioned, spurious relations are expected to show up just by chance. To overcome that problem, we used a random breaking mechanism that chooses a random partition and divides it. However, the signal of spurious relations decreases as the real relations are discovered. This fact motivated the adoption of a cooling down mechanism to decrease the perturbation rate over time. This local search/random perturbation mechanism was successfully applied on several problems and is the basis of algorithms like Simulated Annealing.

Two items need further explanation. The selection of the partition (a) to be merged (Line 3) and the setting of the breaking probability pb. The best results were achieved using a round-robin policy for choosing a and a initial breaking probability $pb = 0.01$, although for noisy problems a greater pb values would produce more accurate results. To cool down, we assign $pb = 0.9pb$.

It is important to notice that decreasing the model building accuracy may have undesired side effects on the ECGA. On the performed tests, the model building accuracy for Algorithm 1 and for Algorithm 2 was the same. This is a general property for trap functions but further experiments are necessary to confirm the hypothesis that both methods have the same accuracy. Initial experiments show that, for problems where proper mixing is fundamental for success (deceptive problems), Algorithm 2 does not suffer from accuracy problems. Challenging problems like noisy problems and exponentially scaled problems might offer more difficulty and further refinements on Algorithm 2 might be necessary.

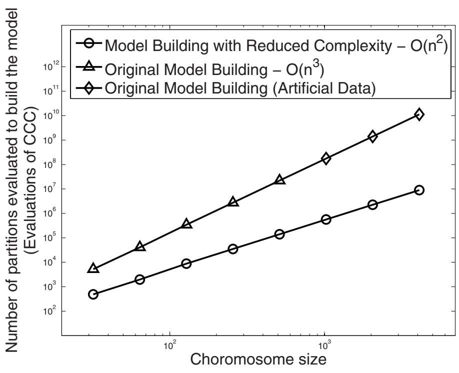


Fig. 1. A comparison of the scalability of the Model Building with reduced complexity and the Original Model Building shows that the proposed approach successfully reduces the number of different partition to be evaluated in one order of complexity, resulting in a speedup greater than 1000 times for individuals of size 4096 bits. The slope of the lines indicates a reduction from cubic to quadratic order.

Figure 1 shows the scalability of the methods on a LogLog scale. On the x axis we have the chromosome size and on the y axis we have the number of evaluations of the CCC necessary to build the correct model. Both the old (original) and the new method are straight lines, what means that they are governed by a power law. The slope of the line for the new method is close to 2 and for the old method close to 3, confirming the scalability hypothesis.

The improvement in the total runtime of the new method when compared to the old one is of order of 10 times for a small problem instance (32 bits) and of more than 1000 times for a 4096 bit problem.

3.2 Improving ECGA through Local Search

GAs are often able to operate over large and multi modal search spaces, presenting a good global search nature. However, local search methods are, in general, more efficient in tuning a solution and reaching the closest local optimum. These two characteristics are desirable and several hybrid approaches have been proposed. There are several ways to hybridize a GA with a local search method and each of them is more suitable to a different class of problems. In this work we propose to use the local search method as a pre-processing mechanism to the ECGA.

The reason for that decision is that the ECGA is very efficient at combining optimal sub-structures, but not so efficient at finding them. To bypass this inefficiency, large population sizes are required to ensure the initial BB supply [3]. Once that initial supply is provided, the ECGA can select promising solutions and learn the appropriate decomposition, building a model that allows it to combine the substructures properly.

The model-building process can become easier if the entropy of the partitions is low, i.e., if we need the smallest number of bits possible to represent that partition. The lowest entropy happens when all non-optimal instances of a BB have probability 0 and only one optimal solution is present. In this case, the entropy is 0.

However, sub-problems may be difficult to solve even if a proper decomposition is known. For instance, a trap function is a difficult function itself. We argue that if it is impossible or too expensive to find the optimal solution to a sub-problem, removing all non-locally-optimal solutions from the population still reduces the entropy significantly[1] and helps the model building process. A local search pre-processing can easily remove such non-locally-optimal solutions.

A local hill climber that processes each variable separately like the one in Algorithm 3, when applied to all individuals in the population, will have the effect of taking every instance of a substructure located on a particular hill on the search space to the peak of that hill. As an effect, all non-locally-optimal solutions will be replaced by a particular local-optimum, reducing the entropy and easing the model building process. Moreover, since fewer different instances of one sub-structure need to be recombined, the overall performance of the ECGA as a mixer of sub-structures will be improved (the mixing time is reduced [18]).

Algorithm 3. A hill climber to be used as a local search for the ECGA

```
1. Given an individual s of length l and fitness f:
2. For each position p in the chromosome:
3.   Flip the bit at p to its complementary value
4.   Evaluate the local change effect and calculate new fitness f'
5.   If f' is not better than f:
6.     Flip the bit p back to the original value
7.   else
8.     f = f'
9. Return the resulting individual
```

This reasoning proved to be true. The application of local search as a pre-processing to the ECGA reduces the population size required by the method, consequently reducing the memory and runtime of the algorithm. Figure 2 compares the minimal population size (determined according to the Bisection Method) required by the ECGA and by the Hybrid to solve problems of growing size (the x axis represents the problem size l). The population size in ECGA scales as $O(l \cdot log(l))$ [9] for problems with constant BB size. Figure 2 shows that the hybrid uses smaller population and scales no worse than the original method.

It is important to remark that the local search procedure is itself time consuming. We can use the locality and independence of the substructures to reduce the cost of the function evaluations required by the hill climber, introducing the concept of a local change evaluation, which evaluation calculates the fitness variation induced by a single bit change. In the mk-trap problem the cost of a local change evaluation is $1/m$ of the cost of a complete function evaluation.

[1] Given that the sub-problem is boundedly multimodal.

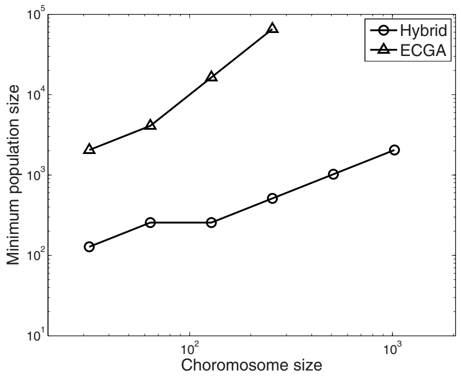


Fig. 2. The Hybridization of the ECGA with a local search procedure reduces the minimum population size required by the algorithm (determined using the bisection method [10]), reducing the memory requirements and potentially the running time

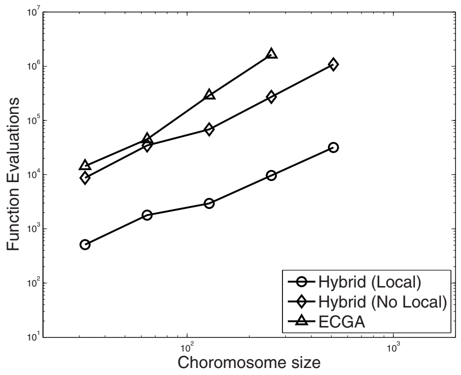


Fig. 3. The Hybridization of the ECGA also affects the number of function evaluations. Considering the cost of a local evaluation as $1/m$ of the cost a complete evaluation we have a significant improvement on the number of function evaluations required. Even when local change evaluations are not available, the number of function evaluations for the hybrid is no worse than the number of function evaluations for the original method. The hybrid still reduces the population size and, consequently, memory and running time.

Figure 3 show the scalability of the number of function evaluations required by the ECGA (ECGA) and the Hybrid. For the latter, two lines where presented, one of them (Hybrid - No Local) assumes that no local change evaluation is available, counting each evaluation as a complete function evaluation. The other line (Hybrid - Local) assumes the availability of local evaluations, counting m of them as one function evaluation. The x axis represents the problem size l.

Figure 3 shows that even in the worst case, the hybrid is no worse than the ECGA. When no local evaluation is available, the hybrid behaves very similar to the ECGA, but requires a smaller population, generating a smaller overhead for model building, selection and other GA related processes, reducing both memory and running time.

4 Future Work

This paper presents enhancements to the ECGA and successfully improves its performance. However, in order to create an effective EA several other enhancements are still necessary. In this section we point and describe some of the next steps toward such effective EA. The proposed enhancements are useful steps, but future work should not be restricted only to these options.

The proposed relaxed model builder shows no accuracy lost for trap functions. Noisy or exponentially scaled problems might be more challenging, requiring further enhancements to the algorithm.

Parallelization: Since the Hill Climber processes each individual independently it is easy to distribute the population over several processors, achieving a speedup near the number of processors for the pre-processing step. Parallelizing the model building deserves special care, since this process centers the information distributed in the population.

Model building improvements: It is possible to further speed-up the algorithm by avoiding unnecessary full model building steps. This can be achieved by three different ways: (I) Sporadic model building [19], which builds the model only after some generations or some important event, instead of every generation; (II) Once and forever model building, which build the model in the first generation and reuses it through the GA run and; (III) Incremental model building, which changes the model using small incremental steps.

Hybridization with competent mutation operator: as discussed in [9], the mutation is useful on deterministic problems, while crossover is useful on noisy problems. A mutation-crossover hybrid can take advantage of both strengths, as showed by [20].

Chromosome compression: compressing the chromosome as in [21] can also improve the performance of the algorithm by reducing the chromosome length and search space.

5 Conclusion

This work presents a proof of principle on the importance of efficiency enhancements for practical GAs. We present two enhancements to the ECGA, a widely used competent genetic algorithm. These enhancements successfully improved performance of the algorithm, with speedups of more than 1000 times for large problems.

The first enhancement, the change on the model building process, was able to reduce significantly the time needed for the model building process. This step represents more than 90% of the original algorithm's runtime. The extension was able to reduce one order of complexity in the model building process, inducing a speedup around 10 times faster for small instance (32 bits) and more than 1000 times faster for a 4096 bit problem.

The second enhancement, the hybridization with a local search preprocessing, successfully reduced the population size required by the algorithm, reducing the memory and runtime requirements. Using the hybrid we were able to solve the same problem with a population at least 32 times smaller. This result holds for small strings (such as 32 bits) and for the larger ones.

Although the results achieved by these enhancements are relevant, this work is only the first step toward a competent and efficient EA, capable of solving difficult large scale problems in practical time. We also pointed some of the next steps toward such EA.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0370, the National Science Foundation under ITR grant DMR-03-25939 at Materials Computation Center and under grant ISS-02-09199 at the National Center for Supercomputing Applications, UIUC. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

1. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
2. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1975)
3. Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Dordrecht (2002)
4. Sastry, K., Goldberg, D., Pelikan, M.: Efficiency enhancment of probabilistic model building genetic algorithm. Technical report, Illinois Genetic Algorithms Laboratory, Univeristy of Illinois at Urbana Champaign, Urbana, IL (2004)
5. Cantu-Paz, E.: Designing Efficient and Accurate Parallel Genetic Algorithms. PhD thesis, University of Illinois at Urbana-Champaign, Illigal Report No 99017 (1999)
6. Goldberg, D.E., Voessner, S.: Optimizing Global-Local Search Hybrids. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 1, pp. 220–228. Morgan Kaufmann, San Francisco (1999)
7. Sinha, A., Goldberg, D.: A survey of hybrid genetic and evolutionary algorithms. Technical report, University of Illinois at Urbana Chapaign, Urbana, IL (1999) IlliGal Report No. 2003004
8. Sinha, A.: Designing efficient genetic and evolutionary algorithm hybrids, Master Thesis, University of Illinois at Urbana Champaign (2003) (IlliGal Report No. 2003020)

9. Sastry, K., Goldberg, D.: Let's Get Ready to Rumble: Crossover Versus Mutation Head to Head. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103. Springer, Heidelberg (2004)
10. Sastry, K.: Evaluation-relaxation Schemes for Genetic and Evolutionary Algorithms. PhD thesis, University of Illinois at Urbana-Champaign (2001)
11. Harik, G., Lobo, F., Goldberg, D.: The compact genetic algorithm. In: Proceedings of IEEE Iternational Conference on Evolutionary Computation (1998), pp. 523–528 (1998)
12. Goldberg, D., Sastry, K., Llorà, X.: Toward routine billion-variable optimization using genetic algorithms: Short Communication. Complexity 12(3), 27–29 (2007)
13. Sastry, K., Goldberg, D., Llora, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 577–584 (2007)
14. Harik, G.: Linkage Learning via probabilistic modeling in the ECGA. Technical report, University of Illinois at Urbana Chapaign, Urbana, IL (1999)
15. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2001)
16. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Proceedings of the Genetic And Evolutionary Computation Conference, pp. 524–532 (1999)
17. de la Ossa, L., Sastry, K., Lobo, F.: χ–ary Extended Compact Genetic Algorithm in C++. Technical report, Illigal Report 2006013, Illinois Genetic Algorithms Lab, University of Illinois at Urbana-Champaign (2006)
18. Thierens, D., Goldberg, D.: Mixing in Genetic Algorithms. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 38–47 (1993)
19. Pelikan, M., Sastry, K., Goldberg, D.: Sporadic model building for efficiency enhancement of hierarchical BOA. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 405–412. ACM Press, New York (2006)
20. Lima, C., Sastry, K., Goldberg, D., Lobo, F.: Combining competent crossover and mutation operators: a probabilistic model building approach. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 735–742 (2005)
21. Yu, T., Goldberg, D.: Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 1385–1392 (2006)

Extreme Value Based Adaptive Operator Selection

Álvaro Fialho[1], Luís Da Costa[2], Marc Schoenauer[1,2], and Michèle Sebag[1,2]

[1] Microsoft Research-INRIA Joint Centre
28, rue Jean Rostand, 91893 Orsay Cedex, France
[2] Team TAO, INRIA Saclay - Île-de-France & LRI (UMR CNRS 8623)
Bât. 490, Université Paris-Sud, 91405 Orsay Cedex, France
`FirstName.LastName@inria.fr`

Abstract. Credit Assignment is an important ingredient of several proposals that have been made for Adaptive Operator Selection. Instead of the average fitness improvement of newborn offspring, this paper proposes to use some empirical order statistics of those improvements, arguing that rare but highly beneficial jumps matter as much or more than frequent but small improvements. An extreme value based Credit Assignment is thus proposed, rewarding each operator with the best fitness improvement observed in a sliding window for this operator. This mechanism, combined with existing Adaptive Operator Selection rules, is investigated in an EC-like setting. First results show that the proposed method allows both the *Adaptive Pursuit* and the *Dynamic Multi-Armed Bandit* selection rules to actually track the best operators along evolution.

1 Introduction

Evolutionary Algorithms (EAs) have demonstrated their ability to solve challenging optimization problems that resisted the standard optimization methods, thanks to their flexibility: EAs can handle structured and mixed search spaces, irregular, noisy, or highly constrained objective functions. However, EAs are still a long way from being part of the standard optimization toolboxes; paradoxical as it may seem, the main reason for that is their high flexibility. Indeed, most EAs provide the user with quite a few levers to tackle problem difficulties; although knowledgeable users can benefit from this diversity and take the most out of the Evolutionary approach, the naive user will generally fail to appropriately tune the EA in a reasonable amount of time. Therefore, a mandatory step for EAs to "cross the chasm"and make it out of the research labs is to offer some automatic parameter tuning capabilities.

Parameter setting was and remains one of the most active research directions in EC (see e.g. [1]). Statistical methods derived from Design Of Experiments have been adapted to the off-line setting of EA parameters [2,3,4,5]; while they are more efficient than classical ANOVA, these methods however require extensive experiments. Online tuning seems another promising way of tackling the

EA parameter control, and in particular, handling the selection of the variation operators. When addressing a new problem, the user can usually define a variety of crossover and mutation operators, fulfilling different roles in the exploitation/exploration dilemma; how to find a strategy for their combined usage is a prominent part of the user's burden. This strategy however most often boils down to a set of static user-defined probabilities, or relative weights, that depend on the user's experience and intuition. After [6], the only dynamic parameter setting strategy widely used in practice concerns the continuous mutation step size adaptation in Evolution Strategies (see references in [6]).

The work presented in this paper is concerned with on-line tuning of operator selection in an EA. Designing an Adaptive Operator Selection (AOS) method involves two main ingredients: the credit assignment mechanism, which associates to each operator a reward, modelling its impact on the progress of evolution; the selection rule, which determines the operator to be used at each time step, depending on the operator rewards. The credit assignment mechanism and the selection rule must be geared to each other to achieve some exploration/exploitation tradeoff in the operator landscape; typically, if the reward provides an instant feedback, modelling the immediate benefits of applying the operator, then the selection rule must ensure that operators with low current benefits can still be explored at a later stage of evolution. Section 2 will present a brief survey of the state of the art, summarizing the credit assignment mechanisms presented in the literature and detailing the selection rules. In particular, the *Probability Matching* (PM) [7] and *Adaptive Pursuit* (AP) [8] will be described, together with the *Dynamic Multi-Armed Bandit* (D-MAB) proposed in [9]. In [9], these three AOS methods have been compared within an artificial setting originally proposed by [8], involving pre-defined rewards whose dynamics are independent from any fitness landscape.

The main contribution of the paper is an original credit assignment mechanism termed *EXtreme value-based Adaptive Operator Selection* (*ExAOS*, described in Section 3), based on empirical order-statistics of the fitness improvement. This mechanism is combined with the above mentioned selection rules and experimentally investigated in an EC-like setting, where the operator rewards computed by *ExAOS* actually follow the dynamics of the evolution trajectory in the fitness landscape (Section 4). This setting considers the eternal OneMax problem; such a simple setting enables to compare the experimental behavior of the online adaptation scheme with the optimal behavior, and to understand the interaction between the dynamics of the fitness landscape, the variance in the operator reward, and the exploration strength in the selection rules. The paper reports on the empirical results in Section 5, and Section 6 concludes with a discussion of the perspectives for further research.

2 Credit Assignment and Adaptive Operator Selection

Credit Assignment. Starting back in the late 80s [10], several methods to assign credit (or reward) to variation operators have been proposed in the literature.

They differ in how they compute the credit of an operator for each newborn offspring. Most methods only use the fitness of the new individual, compared with a reference fitness value: that of the individual parents [11,12,13], of the current best [10] or median [14] individuals. Individuals which do not improve on the reference fitness result in a null credit for the operator. Admittedly, no clear conclusive result can be gathered from those works. Some recent work [15] proposes to use a more sophisticated statistical measure that aims at detecting outliers in the fitness distribution. Reported comparative results with other credit assignment techniques are conclusive, indicating the superiority of this approach over a set of continuous benchmark problems. Though calling to another measure, the method proposed here borrows the idea of detecting beneficial but rare events.

Another distinguishing feature is whether the credit assignment mechanism rewards the operators used to generate the ancestors of the current individual, e.g. using some bucket brigade algorithm [10,14]; creating efficient parents is indeed as important as creating improved offspring. Some authors however do not consider ancestors [11,12] and some even suggest that it sometimes degrades the results [13]. In the rest of the paper, the genealogy of the fit individuals will not be considered. Instant operator credit is computed for each generated offspring, and aggregated through an operator selection rule.

Operator Selection Rules. Most Operator Selection Rules attach a probability to each operator and use a roulette wheel-like process to select the operator to be applied, based on these probabilities[1]. Two such selection rules, namely *Probability Matching* (PM) and *Adaptive Pursuit* (AP), are detailed below.

Let K denote the number of variation operators. Both PM and AP maintain a probability vector $(s_{i,t})_{i=1,K}$, and an estimate of the current operator reward noted $\hat{p}_{i,t}$. At each time t:

• Operator i is selected with probability $s_{i,t}$
• The corresponding reward r_t is computed using the credit assignment at hand
• The reward estimate $\hat{p}_{i,t}$ of the selected operator is updated after r_t, using an additive relaxation mechanism with learning rate α $(0 < \alpha \leq 1)$. It controls the memory of the reward estimate (forgettingness increases with α):

$$\hat{p}_{i,t+1} = (1-\alpha)\hat{p}_{i,t} + \alpha\, r_t \tag{1}$$

Probability Matching, a very popular AOS method [7,11,13], aims to making $s_{i,t}$ proportional to $\hat{p}_{i,t}$, while enforcing a minimal amount of Exploration. More formally, let p_{min} denote the minimal probability of selection of any operator, then:

$$s_{i,t+1} = p_{min} + (1 - K * p_{min}) \frac{\hat{p}_{i,t+1}}{\sum_{j=1}^{K} \hat{p}_{j,t+1}} \tag{2}$$

Note that if some operator gets no reward (respectively the maximal reward) for some time, its expected reward will go to p_{min} (resp. $1 - K * p_{min}$). However,

[1] Methods that recompute those probabilities from scratch from the most recent rewards [14,12] will not be considered here.

this convergence is very slow; experimentally, all mildly relevant operators keep being selected, thus hindering the performance of *Probability Matching* [8].

This drawback is partly addressed by the ***Adaptive Pursuit***: Originally proposed for learning automata, this method follows a winner-take-all strategy, selecting at each time step the operator i^*_t with maximal reward, and accordingly increasing its selection probability:

$$\begin{cases} i^* & = argmax\{\hat{p}_{i,t}\,,\, i = 1 \dots K\} \\ s_{i^*,t+1} & = s_{i^*,t} + \beta\,(1 - (K-1)p_{min} - s_{i^*,t})\,,\ (\beta > 0), \\ s_{i,t+1} & = s_{i,t} + \beta\,(p_{min} - s_{i,t})\,,\ \text{for } i \neq i^* \end{cases} \tag{3}$$

Both PM and AP thus involve the p_{min} parameter to guarantee a sufficient exploration of the operators; AP additionally involves the learning rate β, controlling the greediness of the winner-take-all strategy.

Multi-Armed Bandit Methods. Another approach is inspired from the Multi-Armed Bandit framework, first introduced in the context of Operator Selection by the authors [9]. Multi-Armed Bandit algorithms have been initially proposed as decision making algorithms in uncertain environments.

The so-called *Upper Confidence Bound* (UCB) algorithm devised by Auer et al. [16] achieves the optimal cumulative reward through an Exploration vs Exploitation-based criterion: Let $n_{i,t}$ denote the number of times the i-th arm has been played up to time t, and let $\hat{p}_{i,t}$ denote the average corresponding reward. UCB1 selects in each time step t the arm maximizing:

$$\hat{p}_{j,t} + C\sqrt{\frac{\log \sum_k n_{k,t}}{n_{j,t}}} \tag{4}$$

where C is the *Scaling* factor, controlling the exploration/exploitation tradeoff: the left term in Eq. (4) favors the option with best reward (exploitation) while the right term ensures that each arm is selected infinitely often (exploration). The efficiency of this rule follows from the fact that the lapse of time between two selections of under-optimal arms increases exponentially.

Unfortunately, MABs are not suited to dynamic environments: if the current best option becomes less efficient at some later stage, and happens to be outperformed by another one, it will take a long time before the latter option catches up. A *Dynamic Multi-Armed Bandit* algorithm (D-MAB) was thus proposed in [9], that combines MAB ideas with a specific statistical test known as Page-Hinkley (PH) [17], which is used to detect the changes in the reward distribution, and, upon such a detection, restart the MAB.

More precisely, let $\bar{r}_\ell$ denote the average of $r_1, \dots r_\ell$ and let e_ℓ denote the difference $r_\ell - \bar{r}_\ell + \delta$, where δ is a tolerance parameter. The PH test considers the random variable $m_t = \sum_1^t e_i$. When the difference between $M_t = \max_{i \leq t} m_i$ and m_t is greater than some user-specified threshold γ, the PH test is triggered.

The PH test involves two parameters. Parameter γ controls the trade-off between false alarms and un-noticed changes. Parameter δ enforces the robustness of the test when dealing with slowly varying environments. Following initial experiments in [9], δ was set to 0.15 in all experiments here.

3 *EXtreme Value-Based Adaptive Operator Selection*

This section presents a proposal for Credit Assignment, to be combined with a selection rule to achieve an Adaptive Operator Selection. Let $\mathcal{F}$, o and x respectively denote the fitness function (to be maximized), a variation operator, and an element of the current population. As discussed in Section 2, the proposed credit assignment will only take into account the non-negative fitness differences $(\mathcal{F}(o(x)) - \mathcal{F}(x))_+$. The proposed mechanism is inspired from the following remark. Let us consider an operator bringing frequent small improvements, and compare it with an operator bringing rare large improvements. The latter one will hardly be considered if the reward reflects the *average* fitness improvement, for the average estimated after a few trials is likely to be 0, implying that very few further trials will take place. Hence, in agreement with [15], attention should be payed to extreme, rather than average, events. Incidentally, the role of extreme events in design has long been acknowledged in numerical engineering (e.g. taking into account rogue waves when dimensioning an oil rig); it receives an ever growing attention in the domain of complex systems, as extreme events govern diffusion-based processes ranging from epidemy propagation to financial markets.

The proposed credit assignment mechanism, referred to as *EXtreme value-based Adaptive Operator Selection* (*ExAOS*), proceeds as follows. When operator o is selected after the selection rule under examination (PM, AP or D-MAB), o is applied on the current individual x; the fitness of the offspring is computed and the current improvement is added to the window (*FIFO* order, with window of size W); lastly, the operator reward is set to the maximal fitness improvement in this time window. Formally, let t be the current time step, and t_1 (respectively t_k) denote the time step where operator o was used for the last time (resp., the last time before t_{k-1}). If $\delta(t)$ denotes the fitness improvement observed at time t, then the expected reward for operator o is computed as:

$$\hat{p}_t = argmax\{\delta(t_i), i = 1 \ldots W\} \quad (5)$$

Hence, the *EXtreme value-based Adaptive Operator Selection* mechanism involves a single parameter W, the window size. This parameter W is meant to reflect the time scale of the process; if too large, operators will be applied after their optimal epoch and the switch from the previous best operator to the next best one will be delayed. If W is too small, operators causing large but infrequent jumps will be ignored (as successful events will not be observed at all in the first place) or too rapidly forgotten.

4 Experimental Setting

The artificial setting first proposed in [8] and used in [9] to compare PM, AP (and D-MAB) involved two main simplifications. Firstly, the reward associated to each operator is assumed to be uniform in a given interval. Secondly, the average reward of every operator is subject to abrupt periodic modifications, jumping from one given interval to another.

The experiments below consider a more realistic environment, embedding the *ExAOS* and the adaptive operator selection rules in an actual EA; rewards are computed after the *ExAOS* mechanism, and their dynamics depends on the evolution trajectory and the fitness landscape. It involves the OneMax problem (the "Drosophila of EC"), with $N = 10,000$ bits. Only mutation operators are considered, ranging from the standard bit-flip operator (every bit is flipped with probability $1/N$) to the b-bit mutations (flipping exactly b randomly chosen bits) with $b = 1, 3, 5$. A standard $(1 + \lambda)$-EA is used (λ offspring are created from the current parent; next parent is the best among the current offspring and parent). One main advantage of this setting is to enable the assessment of the approach by comparison with the known optimal behavior.

In many respects, the considered setting is still far from being realistic evolutionarily speaking (applying a $(1 + \lambda)$-EA, $\lambda > 1$ with b-bit mutations is meaningless on the OneMax problem – though it might make more sense on multi-core architectures). It nevertheless confronts the proposed approach with the actual difficulties of taming a dynamic system, where the decisions made govern the expected benefits of further decisions (the selected operators determine the position of the population and hence the improvement expectation of the operators at further stages), as opposed to [8,9]. The considered setting is thus meant to be a "sterile EC-like" environment.

The *ExAOS* mechanism is independently investigated in combination with the three selection rules, AP, PM and D-MAB. The goal of the experiments is to assess the relevance of *ExAOS* in interaction with the three selection rules. The main criterion of performance clearly is the average time-to-solution, though the ability of the adaptive scheme to track the best operator is also considered. In all reported experiments, the initial individual is set to $(0, \ldots, 0)$. However, as PM was found significantly outperformed in all pairwise tests, its results are not presented here. Every selection rule is used with its optimal setting, determined after a preliminary DOE campaign [9]. All results are validated using 11 independent runs and followed by a one-way ANOVA with $\alpha = 0.05$, eventually followed by pairwise Scheffé tests.

5 Experimental Validation

The optimal baseline is provided by the optimal behavior of all operators (computed by a Monte-Carlo simulation). Fig. 1 depicts the operator landscape from the perspective of a $(1 + 50)$-EA; for each fitness we report the fitness gain for the best out of 50 offsprings generated respectively with the 1-,3-,5-bit or bit-flip mutation (averaged on 100 runs).

The trajectory of evolution involves distinct phases. In *stable* phases, the optimal operator remains the same (though its performance might decrease). For instance, while the 5-bit mutation dominates all other operators while $\mathcal{F}(x) <$

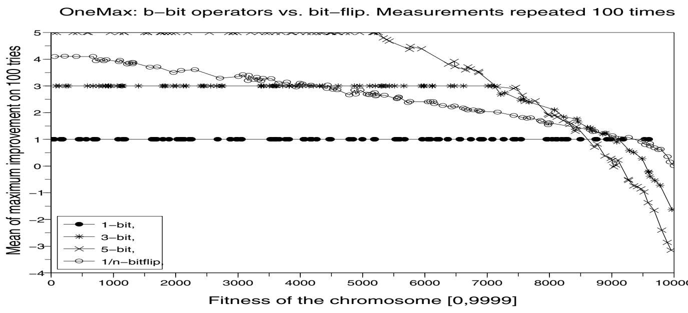


Fig. 1. Average fitness gain with 1-bit, 3-bit, 5-bit and bit-flip mutations within (1 + 50)-EA vs fitness of parent. The best operators are: 5-bit mutation in $[0, 6579]$; 3-bit mutation in $[6580, 8400]$; bit-flip for $[8401, 8600]$; and 1-bit for fitness > 8600.

6579, its performance decreases as the fitness increases after $\mathcal{F}(x) = 5300$. In *transition* phases, the established best operator becomes dominated by another one; the 3-bit mutation outperforms the 5-bit after $\mathcal{F}(x) = 6579$ and the 1-bit mutation outperforms the 3-bit after $\mathcal{F}(x) = 8601$. The last phase is a *desert*, where hardly any operator brings any improvement.

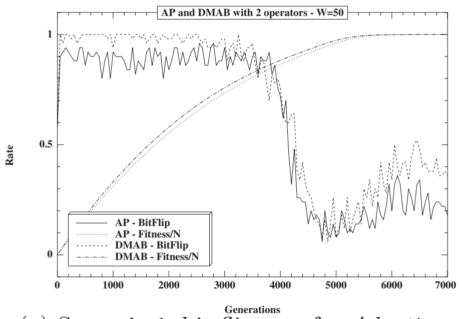
Such an operator landscape enables to assess the basic skills of an Adaptive Operator Selection mechanism: the ability to pick up the best operator and stick to it in stability phases; to swiftly switch to the next best operator in transition phases; and to remain efficient during the desert phases.

Scenario 1. The first experiment considers only the 1-bit and bit-flip mutations, examining the operator rates adapted by *ExAOS* ($W = 50$), AP and D-MAB selection rules, comparatively to the optimal decisions.

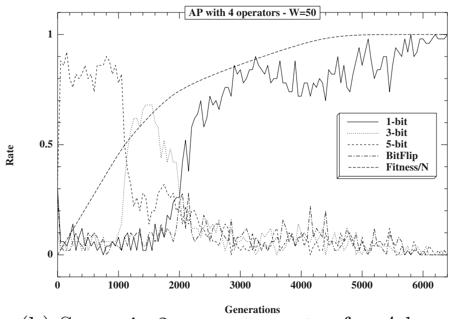
At the beginning of the trajectory (from $(0, \ldots, 0)$), the 1-bit mutation brings a constant improvement of 1 (independently of λ) whereas the average expected reward of bit-flip increases with λ, but with a high variance. This intuition is confirmed by simulations with $\lambda = 1, 5$, and 10. When $\lambda = 1$, bit-flip outperforms 1-bit only until fitness=7, but until fitness=4753 when $\lambda = 5$, fitness=6469 when $\lambda = 10$, and until fitness=8722 when $\lambda = 50$. Therefore, the optimal decision in a $(1 + \lambda)$-EA would be to *always* start with the bit-flip mutation, and to switch to 1-bit afterwards (e.g. at fitness = 8722 for $\lambda = 50$).

The experimental results (Fig. 2.a) demonstrate a good agreement with the optimal rates; the bit-flip rate is close to 1 in the early stages of evolution and switches to p_{min} shortly after the transition point. In the desert phase where rewards are extremely rare, the selection rule consistently selects the 1-bit mutation in the majority of cases, although a high level of exploration is still performed (and would allow any beneficial operator to eventually catch up).

Principled investigations varying λ and W are reported in Table 1, using the best naive strategy (among different fixed mixtures of operators, including using each operator alone) as basline. Firstly. AP and D-MAB obtain comparable



(a) Scenario 1, bit-flip rate for *Adaptive Pursuit* and D-MAB.



(b) Scenario 2. operator rates for *Adaptive Pursuit*

Fig. 2. Adaptive Operator Selection: Operator Rates for $\lambda = W = 50$ (avg. / 11 runs)

Table 1. Comparative results on both scenarios for AP and D-MAB. The figures are the mean number of **generations** (std. dev.) to reach the global optimum. The *best naive* strategy is chosen among 1-bit alone, bit-flip alone or a uniform mixture of both.

		Scenario 1		Scenario 2		Best
λ	W	AP	D-MAB	AP	D-MAB	naive
1	1	93720 (5158)	95296 (6224)	91221 (7738)	$> 100k$	1-bit
	50	93890 (7363)	98871 (3704)	92700 (6800)	98467 (4067)	94928 (4776)
2	1	51629 (2910)	54880 (6379)	49609 (4159)	67085 (7911)	1-bit
	50	52905 (5667)	58514 (6087)	48950 (6352)	59496 (9780)	51817 (3760)
5	1	25284 (1128)	26230 (3096)	21536 (1640)	27421 (2500)	1-bit
	50	24668 (1954)	25683 (1180)	21225 (1776)	24966 (5161)	25715 (1392)
10	1	16558 (980)	16437 (1174)	12769 (1035)	15068 (1230)	1-bit
	50	14521 (1165)	15265 (1011)	12517 (967)	14256 (1748)	16740 (597)
25	1	10285 (326)	10343 (720)	7937 (501)	7778 (591)	Uniform
	50	8830 (493)	8733 (529)	7393 (614)	7728 (768)	10752 (309)
50	1	7882 (245)	7547 (318)	5715 (212)	5786 (364)	Uniform
	50	6619 (285)	6460 (285)	5476 (248)	5513 (431)	7329 (147)

performances on this scenario, not significantly better than the best of the naive strategies: ANOVA rejects the null hypothesis, but all pairwise Scheffé tests fail, even though the means of AP and D-MAB are slightly better than the others.

Secondly, the improvement of using memory ($W = 50$) is found significant for $\lambda \geq 10$ (with decreasing p-values for increasing λ). Indeed, the instant reward ($W = 1$) gives little information on the expected fitness gain out of λ offspring (or even $\lambda/2$ offspring during the exploration phase).

Scenario 2. Scenario 2 presents the AOS mechanism with two additional difficulties. Firstly, it considers all four mutation operators (1,3,5-bit and bit-flip), and thus expectedly requires a higher amount of exploration. Secondly, the operator landscape involves several transition points (Fig. 1): the 5-bit mutation is the best one until $\mathcal{F}(x) = 6579$, the 3-bit mutation until $\mathcal{F}(x) = 8400$, then the bit-flip until $\mathcal{F}(x) = 8722$, and finally the 1-bit dominates until the end. Qualitatively, the experimental results of *Adaptive Pursuit* (Fig. 2.b) and D-MAB (not shown) closely match the above optimal behavior for $W = 50$.

Again, Table 1 reports on the performances of AP and *Dynamic Multi-Armed Bandit* for different values of λ and W. The results are statistically similar to those of scenario 1: ANOVA rejects the null hypothesis, but no pair-wise difference can be found significant between AP, D-MAB and the best naive strategy, even though AP and D-MAB have larger means for large values of λ (AP slightly outperforming here D-MAB). Regarding the effect of memory, setting $W = 50$ does bring, here again, some improvements of the mean performances, but these differences are found significant only in the case of AP with $\lambda = 50$.

6 Discussion and Perspectives

Compared to earlier work related to Adaptive Operator Selection [8,9], this paper presents two extensions. The first one is a new credit assignment method, *EXtreme value-based Adaptive Operator Selection*, translating the fitness gains brought by an operator into actionable rewards, to be exploited by selection rules such as *Adaptive Pursuit* or *Dynamic Multi-Armed Bandit*. Along the same lines as [15], *ExAOS* is driven by the extreme fitness gains brought by an operator, as opposed to the average fitness gain. The *rationale* is that EC must be able to explore "risky" operators, providing rare and large jumps, while average gain-based rewards are strongly biased toward conservative strategies.

The second contribution of the paper is an experimental setting enabling to investigate the AOS behavior *in situ*; although it is still a long way from being evolutionarily challenging, this setting definitely improves on the one considered in [8,9], as it actually couples AOS with an evolutionary-driven system. Within this setting, experiments demonstrate that the adapted operator rates satisfactorily match the optimal ones. However, some problems remain, regarding the (meta-)parameter setting of those methods: The best results of PM and AP were obtained using p_{min}=0, contradicting its definition; Similarly, the performances of D-MAB using *ExAOS* could be improved by a better understanding of the interaction between the scaling factor C (Eq. 4), the PH threshold γ (see Section 2, or [9]), λ and W. Hopefully, the proposed experimental setting will help us in that respect, using other well-known benchmark functions.

Further research is concerned with assessing the respective roles of the time window and the maximum improvement. Furthermore, it has been emphasized that one of EC strengths is to be a rank-based optimization method [18,19]. Accordingly, the reward associated to an operator might consider the top rank of the last fitness gains, as opposed to, its extreme value. Another perspective is to use the extreme value statistics to online adapt the λ parameter in $(1+\lambda)$-EA;

exceptional offspring (wrt extreme gains) can immediately replace the parent without waiting for all λ offspring to be generated.

References

1. Lobo, F., Lima, C., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. Springer, Heidelberg (2007)
2. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) Proc. GECCO 2002, pp. 11–18. Morgan Kaufmann, San Francisco (2002)
3. Yuan, B., Gallagher, M.: Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 172–181. Springer, Heidelberg (2004)
4. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: Proc. CEC 2005, IEEE Press, pp. 773–780. IEEE Press, Los Alamitos (2005)
5. Nannen, V., Eiben, A.E.: Relevance estimation and value calibration of evolutionary algorithm parameters. In: Proc. IJCAI 2007, Hyderabad, India, pp. 975–980 (2007)
6. De Jong, K.: Parameter Setting in EAs: a 30 Year Perspective. In: [1], pp. 1–18
7. Goldberg, D.: Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding. Machine Learning 5(4), 407–426 (1990)
8. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Beyer, H.G. (ed.) Proc. GECCO 2005, pp. 1539–1546. ACM Press, New York (2005)
9. Da Costa, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Keijzer, M., et al. (eds.) Proc. GECCO 2008, ACM Press, New York (to appear, 2008)
10. Davis, L.: Adapting operator probabilities in genetic algorithms. In: Schaffer, J.D. (ed.) Proc. ICGA 1989, pp. 61–69. Morgan Kaufmann, San Francisco (1989)
11. Lobo, F., Goldberg, D.: Decision making in a hybrid genetic algorithm. In: Proc. ICEC 1997, pp. 121–125. IEEE Press, Los Alamitos (1997)
12. Tuson, A., Ross, P.: Adapting operator settings in genetic algorithms. Evolutionary Computation 6(2), 161–184 (1998)
13. Barbosa, H.J.C., e Sá, A.M.: On adaptive operator probabilities in real coded genetic algorithms. In: XX Intl. Conf. of the Chilean Computer Science Society (2000)
14. Julstrom, B.A.: What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm on genetic algorithms. In: Eshelman, L.J. (ed.) Proc. ICGA 1995, pp. 81–87. Morgan Kaufmann, San Francisco (1995)
15. Whitacre, J.M., Pham, T.Q., Sarker, R.A.: Use of statistical outlier detection method in adaptive evolutionary algorithms. In: Cattolico, M. (ed.) Proc. GECCO 2006, pp. 1345–1352. ACM, New York (2006)
16. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47(2/3), 235–256 (2002)
17. Page, E.: Continuous inspection schemes. Biometrika 41, 100–115 (1954)
18. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
19. Gelly, S., Ruette, S., Teytaud, O.: Comparison-based algorithms are robust and randomized algorithms are anytime. Evolutionary Comp. 15(4), 411–434 (2007)

Uncertainty Handling in Model Selection for Support Vector Machines

Tobias Glasmachers and Christian Igel

Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany
{Tobias.Glasmachers,Christian.Igel}@neuroinformatik.rub.de

Abstract. We consider evolutionary model selection for support vector machines. Hold-out set-based objective functions are natural model selection criteria, and we introduce a symmetrization of the standard cross-validation approach. We propose the covariance matrix adaptation evolution strategy (CMA-ES) with uncertainty handling for optimizing the new randomized objective function. Our results show that this search strategy avoids premature convergence and results in improved classification accuracy compared to strategies without uncertainty handling.

1 Introduction

Support vector machines (SVMSs) are powerful algorithms for supervised learning, especially for binary classification [1,2]. However, their performance crucially depends on appropriate model selection, that is, the choice of the right kernel and the right regularization parameter. If a parametrized family of kernel functions is considered, model selection reduces to real-valued optimization. We propose evolution strategies (ES) for solving the resulting optimization problem (see [3,4] and references therein), in particular if the model selection criterion is not differentiable and using grid-search is not possible because of the dimensionality.

Cross-validation is regularly applied as a model selection criterion to estimate the quality of a parameter vector (i.e., as a fitness function). We argue that the cross-validation procedure suffers from its fixed partition of the available data into training and validation sets. Especially for small datasets this has a considerable influence on the objective function and the locations of its minima. Therefore we propose to average over all possible dataset partitions to increase reliability. The resulting fitness function is only of theoretical interest because of the complexity of its computation. We avoid this computational problem by sampling, but at the cost of introducing uncertainty. Another advantage of this averaging is that the performance measure gets more fine-grained when using the 0-1-loss and therefore provides additional information for the search algorithm.

The aim of the present paper is to assess the effects of noise introduced into the SVM model selection due to this sampling. We apply the highly efficient covariance matrix adaptation ES (CMA-ES) to the minimization of the new fitness function [5,6]. Recently, a simple and efficient uncertainty handling mechanism has been proposed for the CMA-ES [7,8], which we employ to handle the uncertainty in our model selection criterion.

The paper is organized as follows. In the next section we introduce the CMA-ES and the noise-handling technique. In Section 3 we briefly review SVMs for binary classification. Then we motivate our model selection objective function. An empirical evaluation is presented in Section 5, and we conclude with a short summary.

2 Handling Uncertainty in Evolution Strategies

We first briefly describe the CMA-ES [5,6,8] and then its extension to adaptive reevaluation for noisy optimization proposed in [7,8].

CMA-ES. In each generation of the CMA-ES λ offspring are generated. Their fitness is evaluated and the $\mu = \lfloor \lambda/2 \rfloor$ best form the next parent population. In each iteration, the kth offspring $\boldsymbol{x}_k \in \mathbb{R}^n$ is created by multi-variate Gaussian mutation and weighted global intermediate recombination: $\boldsymbol{x}_k = \langle \boldsymbol{x}_{\text{parents}} \rangle_{\boldsymbol{w}} + \sigma \boldsymbol{z}_k$, where $\boldsymbol{z}_k \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{C})$ and $\langle \boldsymbol{x}_{\text{parents}} \rangle_{\boldsymbol{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i\text{th-best-parent}}$ ($\boldsymbol{w}_i \propto \ln(\mu + 1) - \ln(i)$, $\|\boldsymbol{w}\|_1 = 1$). The CMA-ES is a variable metric algorithm adapting both the n-dimensional covariance matrix $\boldsymbol{C}$ of the normal mutation distribution as well as the *global step size* $\sigma \in \mathbb{R}^+$. In the basic algorithm, a low-pass filtered *evolution path* $\boldsymbol{p}$ of successful (i.e., selected) steps is stored, $\boldsymbol{p} \leftarrow \eta_1 \boldsymbol{p} + \eta_2 \left(\langle \boldsymbol{x}_{\text{new parents}} \rangle - \langle \boldsymbol{x}_{\text{old parents}} \rangle \right)$, and $\boldsymbol{C}$ is changed to make steps $\boldsymbol{p}$ more likely: $\boldsymbol{C} \leftarrow \eta_3 \boldsymbol{C} + \eta_4 \boldsymbol{p}\boldsymbol{p}^T$ (this rank-one update of $\boldsymbol{C}$ is augmented by a rank-μ update, see [6]). The variables $\eta_1, \dots, \eta_4$ denote fixed learning rates and normalization constants set to default values [8]. The global step size σ is adapted on a faster timescale. It is increased if the selected steps are larger and/or more correlated than expected and decreased if they are smaller and/or more anticorrelated than expected. The highly efficient use of information and the fast adaptation of σ and $\boldsymbol{C}$ makes the CMA-ES one of the best direct search algorithms for real-valued optimization [9].

Uncertainty Handling. Evolution algorithms are well suited for optimization in noisy environments, see [10] for a general overview and [11] for a book on ESs for noisy optimization. The population-based approach, the averaging in the recombination process, and the rank-based, non-elitist selection are inherent features that make the CMA-ES less vulnerable to noise. However, if the signal to noise ratio is too small, special uncertainty handling is required. Here we use a slightly simplified version of the uncertainty handling proposed in [7,8]. It is called *UH-CMA-ES* and relies on adaptive reevaluation of solutions.

Because the selection process is rank-based, we only care about noise if it changes the ranking of offspring. In our scenario, individuals can be reevaluated and computing the mean or median of several evaluations reduces the noise level. However, the signal to noise ratio changes in the course of evolution. Because every fitness evaluation is time consuming, we implement a strategy that adapts the number of evaluations per individual such that individuals are not evaluated too often, but still often enough so that the fitness values can guide the optimization.

We use an algorithm to detect the effective noise by monitoring the stability of the ranking of the offspring. Following [7,8], we consider a population $\mathcal{L}$ composed of two copies of the current offspring population (i.e., each offspring is contained twice in $\mathcal{L}$) and reevaluate λ_{reev} of them. Then we sort $\mathcal{L}$ twice using the new and the old fitness values (f_i^{new} and f_i^{old}, $i = 1\dots,2\lambda$), respectively, and determine the ranks $\text{rank}(f_i^{\text{new}})$ and $\text{rank}(f_i^{\text{old}})$, respectively, of each reevaluated individual x_i. Then we compute the *rank change*

$$\varDelta_i = |\,\text{rank}(f_i^{\text{new}}) - \text{rank}(f_i^{\text{old}})| - 1 \ .$$

The *uncertainty level* s is now defined by

$$s = \frac{1}{\lambda_{\text{reev}}} \sum_{i,x_i \text{was reevaluated}} \Big(2\varDelta_i - \varDelta_\theta^{\text{lim}}(\text{rank}(f_i^{\text{new}}) - \mathbb{I}\{f_i^{\text{new}} > f_i^{\text{old}}\}) \\ - \varDelta_\theta^{\text{lim}}(\text{rank}(f_i^{\text{old}}) - \mathbb{I}\{f_i^{\text{old}} > f_i^{\text{new}}\})\Big) \ .$$

The indicator function $\mathbb{I}$ is one if its argument is true and zero otherwise. The parameter $\theta \in [0,1]$ (here set to 0.2) controls the level of noise we tolerate and $\varDelta_\theta^{\text{lim}}(r)$ denotes the $\theta \times 50$ percentile of the possible rank changes (given by the $2\lambda - 1$ values $|1-r|, |2-r|, \dots, |2\lambda - 1 - r|$) when having the original rank r.

If $s > 0$ we increase the number of evaluations in the computation of a fitness value by a factor of α. Otherwise we decrease the number of evaluations by $1/\alpha$.

The reevaluation is done before the environmental selection in the standard CMA-ES, which uses the median of the fitness values of the reevaluated individuals for ranking. The additional fitness evaluations increase the computational costs per generation. However, we reevaluate on average only $\overline{\lambda}_{\text{reev}} = \max(\lambda/10, 2)$ individuals in each generation.

3 Support Vector Machines

Support vector machines are considered state-of-the art in machine learning for pattern recognition, in particular for binary classification [1,2].

In supervised learning we consider an input space X and an output space $Y = \{-1,1\}$. The learning is driven by sample data $S = \{(x_1,y_1),\dots,(x_\ell,y_\ell)\}$ with $x_i \in X$ and labels $y_i \in Y$ for $1 \le i \le \ell$ drawn independently from some fixed unknown distribution p over $X \times Y$. The goal of binary classification is to infer from S a hypothesis $h : X \to Y$ minimizing the expected loss $R_p(h) = \int_{X\times Y} L(y,h(x))\,\mathrm{d}p(x,y)$ corresponding to the generalization error. We consider the 0-1-loss given by $L(y,h(x)) = (-h(x)y+1)/2$ (i.e., the classification error).

Support vector machines transfer the input data to a feature space and perform linear classification in that space. Given a positive semi-definite kernel function $k : X \times X \to \mathbb{R}$ ($\forall x,x' \in X,\ \forall c_1,\dots,c_m \in \mathbb{R} :\ \sum_{i,j=1}^m c_i c_j k(x,x') \ge 0$), we consider the feature space $\mathcal{H}_k = \text{span}\{k(x,\cdot)\,|\,x \in X\}$ and the function class $\mathcal{H}_k^b = \{f(x) = g(x) + b\,|\,g \in \mathcal{H}_k, b \in \mathbb{R}\}$. We classify according to the sign of a function $f \in \mathcal{H}_k^b$. The decision boundary induced by f is a hyperplane in $\mathcal{H}_k$.

Then the hypothesis generated by a 1-Norm Soft Margin SVM corresponds to a solution of

$$\underset{f \in \mathcal{H}_k^b}{\text{minimize}} \quad \frac{1}{\ell} \sum_{i=1}^{\ell} L_{\text{hinge}}(y_i, f(x_i)) + \frac{\gamma_\ell}{2} \|f\|_k^2$$

where $\gamma_\ell = (\ell C)^{-1}$ and the (semi-)norm $\|.\|_k$ is inherited from $\mathcal{H}_k$ to $\mathcal{H}_k^b$. The loss function is given by $L_{\text{hinge}}(y, f(x)) = \max(0, 1 - yf(x))$. That is, we do not only penalize if a pattern x is classified wrongly (i.e., $yf(x) < 0$), but also if the pattern is too close to the separating hyperplane in the sense that $f(x)$ does not meet the functional target margin (i.e., $|f(x)| < 1$). The parameter $C > 0$ controls the trade-off between the optimization goals of reducing the empirical loss measured by L_{hinge} and the complexity of the hypothesis measured by $\|.\|_k$.

The most frequently used kernel (for $X \subset \mathbb{R}^n$) is the radial Gaussian kernel $k(x, x') = \exp(-\gamma\|x - x'\|^2)$ with a single bandwidth parameter $\gamma > 0$. A standard extension is the Gaussian kernel with feature scaling $k(x, x') = \exp\left(-\sum_{i=1}^{n} \gamma_i (x_i - x_i')^2\right)$, which is also known as automatic relevance detection (ARD) kernel and has as many degrees of freedom as the input space has dimensions. The regularization parameter C and the parameters of the kernel function are called hyperparameters. Their proper selection is the model selection problem for SVMs.

4 A Fitness Function for SVM Model Selection

In this section we introduce a natural objective function for SVM model selection. Because the objective function is impractical to compute we propose a randomized variant, which allows to trade-off accuracy and time to compute the objective function value.

4.1 Hold-Out Sets and Cross-Validation

The probably most simple type of objective function for model selection is the error on a hold-out set. Assume we use a fraction of, for example, 1/5 of the training data as a hold-out set. Then we train a learning machine on the remaining 4/5 of the data and compute the fraction of errors on the hold-out set. This error measure is an unbiased estimate of the generalization error of a machine trained on a dataset of size $(4/5)\ell$ sampled i.i.d. from the data generating distribution p. Usually the optimal parameters for this machine will be quite good for a machine trained on the whole dataset of size ℓ, such that this objective function seems to be a simple and appropriate criterion for model selection. But it turns out that the hold-out error has a high variance, in the sense that it strongly depends on the particular partition of the dataset into training and validation sets. This effect is very pronounced for small datasets and for small hold-out sets. On the other hand, the larger the hold-out set the smaller becomes the remaining training set, and this in turn imposes a larger bias in the estimation of the generalization error, because fewer examples are used for training.

Furthermore, the asymmetry between the roles of the reduced training set and the hold-out set is dissatisfactory.

Cross-validation is a simple procedure which improves on these points. However, the partition of the data into training and hold-out sets remains arbitrary. In a k-fold cross-validation procedure the data S are split into k disjoint subsets $S_1, \dots, S_k$ of roughly equal size. Then for each $i \in \{1, \dots, k\}$ a machine is trained on the dataset $S \setminus S_i$ and the error E_i on the corresponding hold-out set S_i is computed. Finally the total error $\sum_{i=1}^{k} E_i$ is the so-called k-fold cross-validation error. In this procedure the underlying loss function is evaluated exactly once on each training example. In the machine learning literature, common values for the parameter k range from three to ten, and the choice $k = 5$ can be considered a default value [12].

Compared to the simple hold-out error the variance of the generalization error estimate is reduced, but because the data used in the different partitions are of course not independent the reduction of the variance is not by a factor of $\frac{1}{k}$. In general the cost for the computation of the cross-validation error is k times the cost of the computation of the hold-out error. Especially for small datasets the cross-validation error can heavily depend on the partition $S_1, \dots, S_k$ of the dataset and thus has a relatively high variance w.r.t. the choice of the partition. This is an unsatisfactory situation as there is no such thing as a canonical data partition.

4.2 Bootstrapping

Conceptually it is straight-forward to avoid the problem of having an a priori fixed partition of the data. We define the set $J_n = \left\{ I \subset \{1, \dots, \ell\} \;\middle|\; |I| = n \right\}$ of index subsets of size n, leading to the objective function

$$\bar{B} = \frac{1}{|J_n|} \sum_{I \in J_n} \left(\sum_{i \in I^C} L(y_i, h_I(x_i)) \right)$$

where L is a loss function and h_I is the hypothesis constructed from the data indexed by I. Each summand of this objective function computes the hold-out error of the hold-out set $I^C := \{1, \dots, \ell\} \setminus I$, evaluated on the hypothesis h_I. In the style of the k-fold cross-validation procedure we can choose $n = \lfloor \frac{k-1}{k} \ell \rfloor$, and, as usual, we consider $k = 5$ as the default.

This objective function has the conceptual advantage to be completely symmetric w.r.t. the partition of the dataset into training and hold-out set and computes the probably best possible estimate of the generalization error of a machine trained on n examples. It has the disadvantage that the set J_n grows according to $|J_n| = \binom{\ell}{n}$ which is clearly computationally intractable.

Therefore, we introduce the random variable

$$\hat{B} : J_n \to \mathbb{R} \qquad I \mapsto \sum_{i \in I^C} L(y_i, h_I(x_i))$$

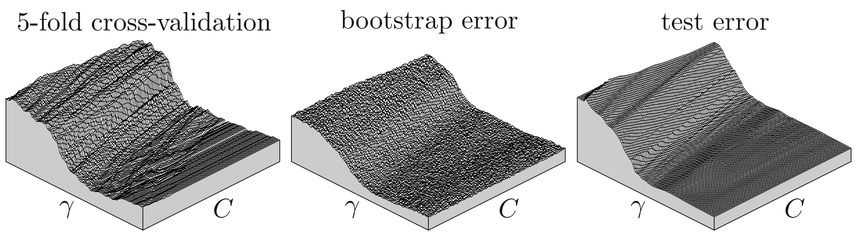


Fig. 1. The plots show the error landscapes for an instance of the `chess` problem with $\ell = 200$ training points (see Section 5) with the radial Gaussian kernel over an equidistant grid on the logarithmic scale for C and γ. Here, the bootstrap error was approximated by averaging over 100 i.i.d. drawn dataset partitions $I \in J_n$ per grid point.

with the uniform distribution for $I \in J_n$. For each fixed index set I we get the hold-out error function $\hat{B}(I)$, which is a function of the SVM hyperparameters. We write $\hat{B}$ for the randomized objective function that picks a random $I \in J_n$ for each of its evaluations. It clearly holds $\mathbb{E}[\hat{B}] = \bar{B}$. This way we are able to avoid a systematic bias resulting from a fixed partition of the data like in the cross-validation procedure, at the cost of a randomized objective function. We will refer to this objective function as the bootstrapping error and use it with the standard 0-1-loss (counting misclassified test patterns).

For the minimization of $\bar{B}$ based on evaluations of $\hat{B}$ we need a search strategy that can deal with a non-differentiable and noisy objective function. This randomized objective function takes as long to evaluate as the simple hold-out error, but we have to be prepared for relatively long optimization runs due to the need for the search algorithm to handle the uncertainty in the objective function evaluations, for example when it is necessary to compute statistics over many evaluation in order to obtain sufficiently reliable information.

Of course there are a lot of possible criteria in between the randomized hold-out error $\hat{B}$, standard cross-validation, and full bootstrapping. We can choose a subset of J_n of considerably smaller size (which should be as symmetric as possible), or take the mean over a few index sets sampled from J_n. The most straight-forward example is to randomly pick a new partition of the dataset for each evaluation of the cross-validation error, which requires a search strategy that can deal with uncertain function evaluations as needed for the minimization of $\hat{B}$. We could even define a (weighted) mean over all choices of n. This leads to a large number of deterministic or randomized objective functions, but for the sake of clarity and for conceptual reasons we stick to the basic randomized hold-out error $\hat{B}$.

The plots in Fig. 1 illustrate the difference between cross-validation and bootstrap error. It is obvious that minimization of the newly proposed bootstrap error gives much more reliable results than cross-validation with a fixed partition of the data.

5 Experimental Evaluation

The focus of our experiments is to assess the effect of uncertainty handling via self-adaptation in the CMA-ES in combination with our new model selection criterion. As discussed above, the UH-CMA-ES algorithm decides automatically how many averages it computes to make its fitness evaluations sufficiently reliable. We compare this strategy to the standard variant of the CMA-ES. Because the standard CMA-ES has no special uncertainty handling, we incorporate the averaging into the objective function simply by computing the statistics over a fixed number of realizations in each evaluation. Thus, we ask for the differences of averaging at the level of the objective function or the search algorithm. Furthermore, we demonstrate that the standard cross-validation procedure indeed suffers from its fixed partition of the data.

5.1 Setup

We consider four experimental setups. The first and most naïve strategy (referred to as `CMA-1`$\times$) is to apply the standard CMA-ES to the randomized bootstrapping objective function $\hat{B}$, ignoring its uncertainty. A second strategy (`CMA-5`$\times$) is to use a fixed average of k evaluations of $\hat{B}$ as a fitness function for the CMA-ES. In the style of cross-validation we use $k = 5$ evaluations in our experiments.[1] The third strategy in the comparison, `CMA-CV`, is 5-fold cross-validation without uncertainty, that is, using a random but fixed partitioning of the data. Again, the CMA-ES is used to minimize the resulting error function. We compare these strategies to the UH-CMA-ES applied to the $\hat{B}$ objective function.

As a proof of concept, the experiments are carried out on four benchmark datasets. In the *chess board problem* we consider the input space $X = [0, 4)^2$ and sample x from the uniform distribution on X. Then we assign a label according to the fixed rule $y = (-1)^{\sum_{i=1}^{2} \lfloor x_i \rfloor}$. This rule assigns labels according to the colors of the fields of a chess board of size 4×4 [13]. This distribution will be referred to as the `chess` problem. We use the radial Gaussian kernel for this problem. The next task is called *sparse coordinate problem* (`sparse` problem for short). To generate a sample we draw $n \in \{1, \dots, 6\}$ uniformly at random and set $S = \{1, \dots, 20\} \setminus \{n\}$. For $n \leq 3$ we assign the positive label $y = +1$, and set $y = -1$ otherwise. Then we randomly remove four more elements from the set S. The final representation is chosen to be $x \in \mathbb{R}^{20}$ with $x_i = 0$ if $x \in S$ and $x_i = 1$ otherwise. We apply the Gaussian ARD kernel to this problem. It should identify the first six coordinates as highly discriminative, while the remaining coordinates provide no useful information. In addition to the artificial distributions `chess` and `sparse`, we consider the benchmark problems `banana` and `image` from the benchmark collection introduced by [14] and apply SVM classifiers with radial Gaussian kernels to these problems.

[1] We could instead use any other number of averages, or use a fixed rule how the number of averages changes over time. However, any such strategy requires problem specific knowledge. Because we aim at a general and automated solution we will assume such expert knowledge to be not available in this study.

The search space for the evolutionary algorithms is a low-dimensional vector space, and we use the parameterization $\log(C)$ and $\log(\gamma)$ (or $\log(\gamma_i)$ for the Gaussian ARD kernel) of regularization and kernel parameters. This allows for unconstrained optimization. All parameters of the CMA-ES are set to default values [8], the initial global step size is set to $\sigma = 1$.

Each strategy is given $100,000$ evaluations of $\hat{B}$. This relatively high (in practice presumably too high) number of evaluations is chosen because it is sufficient for the CMA-ES without uncertainty handling to converge. To generate reliable results, we conducted 1000 trials for all experiments and evaluated the classification performance of the resulting machines with a Mann-Whitney U-test. Of course, this requires that the trials are statistically independent. Due to a lack of data this is impossible to ensure on standard benchmark datasets, because there is no alternative to re-using the same data in each trial. The possibility to sample arbitrary amounts of data and thus to ensure statistical independence is the main motivation for the consideration of artificial test problems such as `chess` and `sparse`. All four methods in the comparison are reasonable strategies for SVM model selection. Therefore we expect the differences to be small. Furthermore, the fitness function and the function used to judge the final parameters differ. The objective function of model selection is, of course, the generalization error, which is estimated by the test error.

Table 1. Absolute and relative performance of the classifiers resulting from the parameters found by the different strategies. The errors are given as 25%, 50%, and 75% quantiles over 1000 trials. The comparison matrix on the right uses the symbols $<$, $\ll$, and $\lll$ to indicate that the method in this row performs significantly better than the method in this column with significance levels 0.05, 0.01, and 0.001, respectively. A one-sided Mann-Whitney U-Test (also known as Wilcoxon rank sum test) is used for the comparison. Analogously, the symbols $>$, $\gg$, and $\ggg$ indicate that the method in that row performs significantly worse with the corresponding significance level. If the differences are not significant at a level of at least 0.05 the significance level is reported. Note that for the fixed size datasets `banana` and `image` the trials are not independent, such that the "true" significance levels are in general worse.

	`chess`							`sparse`						
method	q_{25}	q_{50}	q_{75}	(1)	(2)	(3)	(4)	q_{25}	q_{50}	q_{75}	(1)	(2)	(3)	(4)
(1) `CMA-1`×	0.149	0.168	0.187	—	$\ggg$	$\gg$	$\ggg$	0.257	0.274	0.295	—	$\ggg$	$\ggg$	$\ggg$
(2) `CMA-5`×	0.146	0.162	0.181	$\lll$	—	0.13	$\gg$	0.250	0.262	0.281	$\lll$	—	0.44	$\ggg$
(3) `CMA-CV`	0.147	0.163	0.183	$\ll$	0.87	—	$\ggg$	0.251	0.263	0.278	$\lll$	0.56	—	$\ggg$
(4) `UH-CMA`	0.143	0.159	0.178	$\lll$	$\ll$	$\lll$	—	0.249	0.258	0.274	$\lll$	$\lll$	$\lll$	—
	`banana`							`image`						
method	q_{25}	q_{50}	q_{75}	(1)	(2)	(3)	(4)	q_{25}	q_{50}	q_{75}	(1)	(2)	(3)	(4)
(1) `CMA-1`×	0.126	0.138	0.158	—	$\ggg$	$\ggg$	$\ggg$	0.119	0.133	0.154	—	$\ggg$	$\ggg$	$\ggg$
(2) `CMA-5`×	0.124	0.135	0.149	$\lll$	—	0.62	$>$	0.116	0.129	0.144	$\lll$	—	0.52	0.38
(3) `CMA-CV`	0.124	0.134	0.149	$\lll$	0.38	—	$>$	0.116	0.129	0.144	$\lll$	0.48	—	0.34
(4) `UH-CMA`	0.123	0.132	0.147	$\lll$	$<$	$<$	—	0.116	0.129	0.145	$\lll$	0.62	0.66	—

In contrast, we have no alternative to fitness functions computable from the available training data. This difference between the functions used for training and for evaluation is an additional source of perturbations in the results. These two reasons make clear that we need a relatively large number of trials in order to obtain statistically significant results. We used training datasets of size $\ell = 100$. For the artificial problems we sampled test sets of size $100,000$, giving extremely reliable estimates of the generalization error. For the fixed size benchmark problems we used the remaining examples for testing, which amounts to $5,000$ test examples for the `banana` benchmark and $2,210$ for the `image` problem.

5.2 Results and Discussion

The results are summarized in Table 1. The significant differences between the methods clearly indicate that averaging over several evaluations of $\hat{B}$ improves the solution. The CMA-ES profits from automatic uncertainty handling. The UH-CMA-ES method performs clearly best, although it evaluates only a comparatively small number of search points. For the SVM model selection problem, and in particular for the fine-tuning of the SVM hyperparameters, the reliable evaluation of a small number of candidates turns out to be more successful than the cheap but unreliable evaluation of a large number of search points. Furthermore, the experiments indicate that the robust estimation of the generalization error requires a large number of averages over simple hold-out error evaluations.

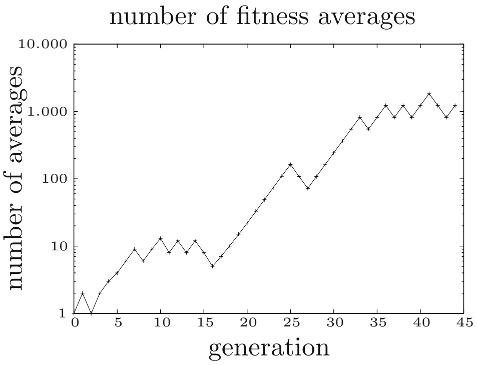


Fig. 2. Typical evolution of the number of averages over the generations of the UH-CMA-ES. Usually only few generations can be evaluated with a limit of 100,000 evaluations of $\hat{B}$.

The plot in Fig. 2 clearly reveals that there is no uniformly best number of averages for all search points, but that the number of averages grows to large numbers if needed. This result is not surprising. Of course, the closer the CMA-ES comes to a local optimum, the worse gets the signal-to-noise ratio. This drives the UH-CMA-ES algorithm to large numbers of averages in late generations. The algorithm very quickly identifies the region of well-generalizing classifiers, and then gradually switches over to fine-tuning of the hyperparameters which requires a large sample per individual.

In our experiments, the bootstrap error $\bar{B}$ is clearly superior to the cross-validation error if the uncertainty of $\hat{B}$ is handled properly.

6 Conclusion

We applied the CMA-ES with and without uncertainty handling mechanism to the problem of model selection for SVMs. As a new model selection criterion, we proposed the minimization of the bootstrapping error $\bar{B}$ based on evaluations of its estimate $\hat{B}$. There are good arguments to prefer this objective function over standard cross-validation. Our experiments support these theoretical considerations and show the advantage of automatic uncertainty handling for this problem. The small overhead of the uncertainty handling for the re-evaluation of some individuals is clearly justified by the resulting improvement in performance.

References

1. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20, 273–297 (1995)
2. Evgeniou, T., Pontil, M., Poggio, T.: Regularization networks and support vector machines. Advances in Computational Mathematics 13, 1–50 (2000)
3. Friedrichs, F., Igel, C.: Evolutionary Tuning of Multiple SVM Parameters. Neurocomputing 64, 107–117 (2005)
4. Mersch, B., Glasmachers, T., Meinicke, P., Igel, C.: Evolutionary Optimization of Sequence Kernels for Detection of Bacterial Gene Starts. International Journal of Neural Systems 17, 369–381 (2007); Selected paper of ICANN 2006
5. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9, 159–195 (2001)
6. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11, 1–18 (2003)
7. Hansen, N., Niederberger, A.S.P., Guzzella, L., Koumoutsakos, P.: Evolutionary optimization of feedback controllers for thermoacoustic instabilities. In: Morrison, J.F., Birch, D.M., Lavoie, P. (eds.) IUTAM Symposium on Flow Control and MEMS. Springer, Heidelberg (2008)
8. Hansen, N., Niederberger, A.S.P., Guzzella, L., Koumoutsakos, P.: A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Transactions on Evolutionary Computation (in press, 2008)
9. Beyer, H.G.: Evolution strategies. Scholarpedia 2, 1965 (2007)
10. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments—a survey. IEEE Transactions on Evolutionary Computation 9, 303–317 (2005)
11. Arnold, D.V.: Noisy Optimization With Evolution Strategies. Kluwer Academic Publishers, Dordrecht (2002)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, Heidelberg (2001)
13. Glasmachers, T., Igel, C.: Gradient-based Adaptation of General Gaussian Kernels. Neural Computation 17, 2099–2105 (2005)
14. Rätsch, G., Onoda, T., Müller, K.R.: Soft Margins for AdaBoost. Machine Learning 42, 287–320 (2001)

Niche Radius Adaptation with Asymmetric Sharing

Vincent van der Goes[1], Ofer M. Shir[1], and Thomas Bäck[1,2]

[1] Leiden Institute of Advanced Computer Science
Universiteit Leiden
Niels Bohrweg 1, 2333 CA Leiden
The Netherlands
[2] NuTech Solutions
Martin-Schmeisser-Weg 15
44227 Dortmund
Germany

Abstract. In the field of *Genetic Algorithms*, *niching* techniques have been invented with the aim to induce *speciation* on multimodal fitness landscapes. Unfortunately, they often rely on a problem-dependent *niche radius* parameter. This is the *niche radius problem*. In recent research, the possibilities to transfer niching techniques to the field of *Evolution Strategies* (ES) have been studied. First attempts were carried out to learn a good value for the niche radius through *self-adaptation*. In this paper we introduce a new niching method for ES with self-adaptation of the niche radius: *asymmetric sharing*. It is a form of *fitness sharing*. In contrast to earlier studies, it does not depend on coupling the niche radius to other strategy parameters. Experimental results indicate that asymmetric sharing performs well in comparison to traditional sharing, without relying on problem-dependent parameters.

1 Introduction

In the history of *Evolutionary Algorithms* (EAs), there have been many attempts to promote population diversity. Sometimes the goal was not just to maintain a higher degree of diversity, but to set off a process of *speciation*, where a multitude of different species evolve simultaneously, as seen in nature. Techniques developed with this goal in mind became known as *niching methods*. They allow an EA to find and maintain multiple local optima, an outcome which is normally prevented by loss of diversity caused by *genetic drift* [7], [12]. One of them is *fitness sharing*. It is was first introduced by Holland [6], and further developed by Goldberg and Richardson [8].

Niching stems from the *ecological niche* concept from biology. A biological niche relates to a complex set of factors that allow the survival of species adapted to it. It includes environmental factors, available resources and behavioral patterns.

In an EA there are no resources, nor is there a direct equivalent of niches. There is only an objective function, mapping an abstract phenotype to an abstract fitness value. The niching approach, however, is to regard the local optima of a multimodal objective function as niches.

Fitness sharing promotes diversification by adjusting fitness values. The idea is to treat fitness as a resource and force all individuals in a niche to share the fitness of the niche with each other. The *raw* fitness of each individual a, $f(a)$, is divided by its *niche count*, m, as in Eq. 1. The niche count of an individual is a measure of the number of individuals that are located in its proximity, that can be considered as its niche. The resulting adjusted fitness value, denoted as $f'(a)$, is called the *shared fitness* of individual a.

$$f'(a) = \frac{1}{m(a)} \cdot f(a) \tag{1}$$

Shared fitness now becomes the new criterion for selection. Thereby, individuals in overpopulated niches are heavily penalized and migration toward more vacant niches is stimulated. However, there is no direct way to measure the number of individuals occupying a niche. The niche count of an individual can only be estimated. This can be carried out by the utilization of a *sharing function*. A sharing function receives the distance between two individuals as input and returns their *sharing value*, which is the degree to which they must share their fitness. The most commonly used sharing function is the *triangular sharing function*, as shown in Eq. 2. The sharing function relies on a threshold distance σ_{share}, referred to as the *sharing radius* or the *niche radius*. It is a parameter of the algorithm, and roughly speaking it corresponds to the expected distance between the niches.

Let d be the distance between two individuals, which is either genotypic- or phenotypic-based, then the triangular sharing function is explictly given by:

$$sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right) & \text{, if } d < \sigma_{share}; \\ 0 & \text{, otherwise.} \end{cases} \tag{2}$$

The total niche count, estimating the population of the niche of an individual, is then defined as the sum of all of its sharing values:

$$m(a) = \sum_{i \in P} sh\left(d(a, i)\right) \tag{3}$$

The problem of finding the optimal value for σ_{share} in fitness sharing, the value which results in the maximal number of maintained niches, is known as the *niche radius problem*.

The standard approach is to assume an even distribution of the peaks over the search space, and then the value of σ_{share} can be approximated from the expected number of peaks (see, e.g., [9]). However, the number of expected peaks is also generally unknown.

Fitness sharing so far has mainly been applied in the field of *Genetic Algorithms* (GAs). In the work presented here fitness sharing will be applied to

Evolution Strategies (ES). An overview of Evolution Strategies can be found in the literature [1,2,3,4]. At the same time, the niche radius problem will be addressed in this paper. Section 2 provides the reader with an overview of the work in this area of research so far. Section 3 introduces a new sharing scheme designed for ES, namely *asymmetric sharing.* In section 4, asymmetric sharing is applied to a test-bed of artificial benchmark problems, and its performance is evaluated. In addition, the same Evolution Strategy is tested with a traditional form of fitness sharing, *continuously updated sharing* [10], as a reference for comparison. The results are summarized and their implications are discussed in section 5. Finally, section 6 offers a brief summary and points out possible directions for future work.

2 Related Work

One way to handle the niche radius problem is to use a clustering scheme instead of a sharing function [11]. A clustering scheme is more flexible than the standard approach, since it does not assume an even distribution of peaks over the search space. However, clustering schemes still depend on problem specific parameters.

A different approach was the development of algorithms that learn the locations of the niches during the course of the evolution. Two such algorithms are *dynamic niche sharing* [13] and *coevolutionary shared niching* [14].

The *ES dynamic niching algorithm* [16] was inspired by dynamic niche sharing. It had a similar ability to identify the locations of niches dynamically. An interesting extension to the ES dynamic niching algorithm was to replace the global, fixed niching radius with an individual, self-adapted niching radius [17]. It was coupled to the self-adapted global step-size, which governs the mutation strength. However, this algorithm introduces a new problem-dependent parameter α. A further refinement was to replace the standard, Euclidian distance measure with the Mahalanobis metric [18].

3 Asymmetric Sharing

3.1 An Additional Strategy Parameter

The basic idea of asymmetric sharing is to add an individual sharing radius to the genetic code as an independent new strategy parameter, which will be denoted as ν. It is not coupled to any other existing strategy parameter. An individual a can now be denoted as:

$$a = (\boldsymbol{x}_a, s_a, \nu_a), \tag{4}$$

where $\boldsymbol{x}_a$ are the object variables and s_a is the set of strategy parameters that are responsible for the variation operators. The new strategy parameter ν requires its own update rules for recombination and mutation. Let P denote a set of ρ parent individuals, let a' be a recombined individual and let a'' be a mutated individual. We propose the following update rules for individual sharing radii:

$$\nu_{a'} = \left(\prod_{i \in P} \nu_i\right)^{\frac{1}{\rho}}, \tag{5}$$

$$\nu_{a''} = \nu_{a'} e^{N\left(0,\tau^2\right)}, \tag{6}$$

where τ is the learning parameter, which will be set to $\frac{1}{\sqrt{n}}$, and where n is search space dimensionality.

The update rule of the mutant in Eq. 6 is generally referred to as *lognormal mutation*. It allows exponential growth or decay of the niche radius, so that individuals can adapt themselves quickly.

The update rule for the recombinant in Eq. 5 is known as *geometric recombination*. As argued by Hansen in [5], when using lognormal mutation, using geometric recombination for the same parameter avoids a bias toward either increasing or decreasing values of that parameter over time.

3.2 Inner and Outer Niche Count

Upon the introduction of an individual niche radius, the question turns up how to calculate the niche count m. The sharing function can no longer rely on a global, fixed radius σ_{share}. Instead, the sharing function now takes an additional parameter, ν. The triangular sharing function in Eq. 2 is transformed into:

$$sh(d,\nu) = \begin{cases} 1 - \left(\frac{d}{\nu}\right) & \text{, if } d < \nu; \\ 0 & \text{, otherwise.} \end{cases} \tag{7}$$

In standard fitness sharing, there exists an implicit symmetry. Since $d(a,b) = d(b,a)$, we also have $sh(d(a,b)) = sh(d(b,a))$. That made it possible to associate a single, unique sharing value with every pair of individuals (a,b). However, the introduction of individual niche radii makes it no longer possible. A pair of individuals a and b is now associated with two different sharing values, $sh(d(a,b),\nu_a)$ and $sh(d(a,b),\nu_b)$: the symmetry is thus broken.

In order to find a way out of this dilemma, consider the following two new types of niche count: the *inner niche count*, corresponding to the niche count of individual a as measured by ν_a, versus the *outer niche count*, corresponding to the niche count as measured by all of the other radii.

$$\begin{aligned} m_{in}(a) &= \textstyle\sum_{i \in P} sh\left(d(a,i),\nu_a\right) \\ m_{out}(a) &= \textstyle\sum_{i \in P} sh\left(d(a,i),\nu_i\right) \end{aligned} \tag{8}$$

If we choose the inner niche count as the new niche count to calculate shared fitness, there will be no stimulus to evolve a niche radius greater than zero. A large niche radius would only reduce fitness. However, should we choose to use the outer niche count to calculate shared fitness, a large niche radius would be a pure advantage in the evolution. The greater the niche radius of an individual, the more the niche counts of competing individuals are increased.

We present *asymmetric sharing* here as an attempt to balance these conflicting forces. In asymmetric sharing, the niche count used for the evaluation of the shared fitness is a weighted average of the inner niche count and the outer niche count:

$$m(a) = w_{in} \cdot m_{in}(a) + w_{out} \cdot m_{out}(a) \tag{9}$$

Now the weights w_{in} and w_{out} are chosen in such a way, that the niche radius of a contributes as much to the niche count of a as it contributes, on average, to the niche count of other individuals. Formally, they are chosen to satisfy the following equation:

$$\frac{w_{out} \cdot \sum_{i \in (P \setminus \{a\})} sh(d(a,i), \nu_a)}{|(P \setminus \{a\})|} = w_{in} \cdot \sum_{i \in (P \setminus \{a\})} sh(d(a,i), \nu_a) \tag{10}$$

The left-hand side of Eq. 10 is the average contribution of ν_a to the niche count of other individuals. The right-hand side is the contribution of ν_a to the niche count of a itself. With the sum of the weights normalized to 1, eq. 10 can be solved for w_{in} and w_{out}, yielding:

$$\begin{aligned} w_{in} &= \tfrac{1}{|P|} \\ w_{out} &= 1 - \tfrac{1}{|P|} \end{aligned} \tag{11}$$

Substituting these values into Eq. 9 results in the final niche count calculation:

$$m(a) = \frac{1}{|P|} \cdot m_{in}(a) + \left(1 - \frac{1}{|P|}\right) \cdot m_{out}(a) \tag{12}$$

3.3 Bottom-Up Sharing

Evolution Strategies typically employ truncation selection, which is a special case of *tournament selection*. As shown in [10], a naive combination of fitness sharing and tournament selection leads to rapid loss of diversity. A special tournament selection scheme, *continuously updated sharing*, avoids the problem. Individuals are selected for survival one by one, updating niche counts in each step.

However, continuously updated sharing interferes with the niche count calculation of asymmetric sharing in a harmful way. The problem is that prior to selection of an individual, its niche radius only affects its own niche count. Its contribution to the niche count of competitors does not come into effect before selection.

Therefore we propose to use a variant of continuously updated sharing, *bottom-up sharing*. It was first introduced in [15]. The shared fitness of each candidate individual is calculated. Then the worst one is eliminated from the population and the remaining population stops sharing with it. The individuals are compared on their new shared fitness and the process is repeated until μ individuals remain.

When implementing asymmetric sharing with bottom-up sharing, special care has to be taken to keep the time complexity at $\mathcal{O}(n^2)$. This can be realized by storing the total outer and inner niche count for each individual at each step. Both the niche counts and the weights are continuously updated. The resulting algorithm is given as Algorithm 1.

Algorithm 1. Asymmetric sharing in pseudocode.

```
do while $|P| > \mu$
  for each $i \in P$ do
    $m_i \Leftarrow \frac{1}{|P|} m_{in,i} + (1 - \frac{1}{|P|}) m_{out,i}$
  od
  worst $\Leftarrow$ individual with lowest shared fitness in P
  $P \Leftarrow P \setminus \{worst\}$
  for each $i \in P$ do
    $m_{in,i} \Leftarrow m_{in,i} - sh(d(i, worst), \nu_i)$
    $m_{out,i} \Leftarrow m_{out,i} - sh(d(i, worst), \nu_{worst})$
  od
od
```

3.4 Asymmetric Restricted Mating

In niching, when two individuals from different niches are recombined into a highly unfit hybrid, this is know as *lethal recombination*. In the context of evolution strategies, lethal recombination will not only be caused by bad recombination of object parameters, as usual. Inconsistent strategy parameters are a second cause of lethal recombination. For asymmetric sharing, the inherited niche radius ν of a hybrid is likely to be inconsistent with the object parameters. So there is good reason to expect a high degree of lethal recombination.

If an Evolution Strategy with comma selection suffers from a high degree of lethal recombination, the algorithm can become unstable, since comma selection requires at least μ offspring with high fitness in each generation. A restricted mating scheme has been shown to be an effective way to prevent lethal recombination in GA's [9]. A restricted mating scheme is proposed here for asymmetric sharing: *asymmetric restricted mating*.

A single random individual a is selected for mating. Now the *set of mutual attraction* of a, $\mathcal{M}(a)$, is constructed. It contains all individuals b (including a itself) at a distance smaller than both niche radii, ν_a and ν_b. From the set of mutual attraction, ρ individuals are drawn at random for recombination. If the size of $\mathcal{M}(a)$ is smaller than ρ, the whole set is used for recombination.

Unfortunately, restricted mating also has a drawback. While it reduces lethal recombination, it restricts exploration of the search space. As Mahfoud stated in [12], page 50: "both interspecies and intraspecies crosses may be beneficial."

Therefore, in this paper a direct trade-off between standard mating and a restricted mating scheme is used: *semi-restricted mating*. When a new individual is created, restricted mating is applied with a fifty percent chance. Otherwise, the standard mating scheme is applied.

4 Experiments

The methods and techniques proposed so far are tested here on four fitness sharing algorithms. Each of them is a standard $(\mu/\rho, \lambda)$-ES with correlated mutations (see, e.g., [3,4]). The first algorithm, **CUS**, employs continuously updated sharing with a fixed niche radius, as proposed in [10]. The second algorithm, **AS**, employs asymmetric sharing (with the bottom-up sharing scheme), as proposed in section 3. The third algorithm, **ASR**, employs both asymmetric sharing as well as the asymmetric restricted mating scheme, as proposed in section 3.4. The fourth algorithm, **ASSR**, is identical to **ASR**, except that it employs semi-restricted mating, as proposed in section 3.4.

In all of these algorithms, the number of recombinant parents is set to $\rho = 2$. Initially, the population is distributed randomly with a uniform distribution over the search space. Mutative step-sizes are all initialized at $\sqrt{n} \cdot l/3$, where n is the problem dimension and l is the diameter of the search space. Rotation angles are initialized randomly, subject to uniform distribution, in $[-\pi, \pi]$. Niche radii are initially set to $\sqrt{n} \cdot l/3$. The evolution is stopped after a fixed number of generations.

The set of test cases contains five maximization problems: *M1* to *M5*. Of these, *M1* and *M2* have a uniform distribution of local optima. Their main purpose is to investigate if asymmetric sharing is able to overcome the lack of a priori knowledge in comparison to sharing with a fixed radius. The other functions have a clearly non-uniform distribution of local optima. The purpose of these functions is to test if asymmetric sharing can improve on the performance of sharing with a fixed radius, in such cases. The maximum number of generations, the values of σ_{share} for **CUS**, as well as μ and λ, are given in Table 1. The value of σ_{share} was set by assuming an even distribution of the niches over the search space. The one exception is *M3*, where the value of σ_{share} has been tuned manually. Some functions are tested multiple times, with different dimensionality n.

Function *M3* has the most widely varying niche distances and sizes among these test functions. The widest peak is about ten thousand times wider than

Table 1. The set of test functions and their associated problem dependent parameter settings. σ_{share} is only used by the **CUS** algorithm.

Name	function	n	domain	niches	μ	λ	maxGen	σ_{share}
$M1_1$	$\prod_{i=1}^{n} \sin(5\pi x_i)^6$	1	$[0,1]^n$	5^n	15	100	20	0.2
$M1_{30}$	$\prod_{i=1}^{n} \sin(5\pi x_i)^6$	30	$[0,1]^n$	5^n	30	210	200	0.2
M2	$\exp(-2\log(2)(\frac{x-0.1}{0.8})^2)\sin(5\pi x)^6$	1	$[0,1]$	5	15	100	50	0.2
M3	$\sin^6(\log_{1.2}(x))$	1	[0.01,100]	16	50	350	500	1
$M4_1$	$\sum_{i=1}^{10} \frac{1}{c_i+(k_i(x-a_i))^2}$	1	$[0,10]^n$	8	30	210	100	1.25
$M4_2$	$\sum_{i=1}^{10} \frac{1}{c_i+\sum_{j=1}^{2}(k_i(x_j-a_i))^2}$	2	$[0,10]^n$	8	30	210	100	1.76
$M4_5$	$\sum_{i=1}^{10} \frac{1}{c_i+\sum_{j=1}^{5}(k_i(x_j-a_i))^2}$	5	$[0,10]^n$	8	30	210	100	2.80
M5	$\sum_{i=1}^{10} \frac{1}{c_i+\sum_{j=1}^{4}(x_j-T_{ij})^2}$	4	$[0,10]^n$	10	30	210	100	5.62

the smallest peak. This makes it especially attractive to investigate the niche radius problem.

Function *M4* is a Shekel function suggested in [19]. All its local optima are located on a line. Function *M5* is a four dimensional Shekel function, for which this is not the case. The values of the matrix $\boldsymbol{T}$ and the vector $\boldsymbol{c}$ in *M5* can be found in the appendix.

Each test involves 100 runs of the algorithm on a given test problem. The quality measured is the number of niches that has been attained in the end of the run, averaged over all runs. A niche is considered to be 'attained' if at least one individual is located in the local region of at least 80% fitness.

5 Results and Discussion

The average final results are summarized in Table 2.

Table 2. Number of peaks attained, aver over 100 runs

function	**CUS**	**AS**	**ASR**	**ASSR**
$M1_1$	5.00	4.97	4.79	4.82
$M1_{30}$	29.65	23.48	1.31	2.49
M2	4.65	3.91	3.41	3.53
M3	8.62	8.90	14.86	15.82
$M4_1$	6.93	7.80	7.14	7.13
$M4_2$	4.70	5.17	5.41	5.34
$M4_5$	1.00	1.95	3.30	3.32
M5	5.79	6.01	6.41	7.54

Table 2 shows that, first of all, **AS** and variants have a reasonable niching capability, without needing a fixed niche radius. The number of maintained peaks does, on average, not collapse to one on any of the test functions. On the functions with uniform niche distributions, *M1* and *M2*, **AS**, the algorithm without any mating restriction, approaches the performance of **CUS** most closely. The data suggests that restricted mating is not an advantage on this type of function.

On the functions with non-uniform niche distributions, however, restricted mating yields more promising results. When comparing the strictly restricted mating scheme of **ASR** to the semi-restricted mating scheme of **ASSR**, the latter comes out best quite convincingly. This supports the hypothesis that finding some kind of balance between interspecies mating and intraspecies mating is desired.

6 Summary and Outlook

In this paper we introduced assymetric sharing as a new variant of niching algorithms in ES, which aims at treating the niche radius problem. The new methods avoid the need for a niche radius parameter or similar parameters to be set.

The behavious of those variants have been tested on a number of benchmark functions. The results indicate that assymetric sharing has the potential to allow niching on certain classes of fitness landscapes. The results look especially promising on functions where the niches vary dramatically in size.

Future research will be required to investigate how the approach behaves in combination with more advanced, derandomized forms of ES and to test it on higher dimensional landscapes, artificial as well as real-world based.

References

1. Rechenberg, I.: Evolutions strategies: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart (1973)
2. Schwefel, H.-P.: Numerical Optimization of Computer Models. Wiley, Chichester (1981)
3. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
4. Beyer, H.-G., Schwefel, H.-P.: Evolution Strategies: A Comprehensive Introduction. Journal Natural Computing 1(1), 3–52 (2002)
5. Hansen, N.: Verallgemeinerte individuelle Schritweiteregelung in der Evolutionsstrategie. PhD thesis, Technical University of Berlin (1998)
6. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor (1975)
7. de Jong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)
8. Goldberg, D.E., Richardson, J.: Genetic Algorithms with Sharing for Multimodal Function Optimization. In: Proceedings of ICGA 1987, pp. 42–50. Morgan Kaufmann, San Francisco (1987)
9. Deb, K., Goldberg, D.E.: An Investigation of Niche and Species Formation in Genetic Function Optimization. In: Proceedings of ICGA 1989, pp. 42–50. Morgan Kaufmann, San Mateo (1989)
10. Oei, C.K., Goldberg, D.E., Chang, S.J.: Tournament Selection, Niching, and the Preservation of Diversity. IlliGAL Report 91011, University of Illinois at Urbana-Champaign (1991)
11. Yin, X., Germay, N.: Improving Genetic Algorithms with Sharing through Cluster Analysis. In: Proceedings of ICGA 1993, pp. 100–101. Morgan Kaufmann, San Francisco (1993)
12. Mahfoud, S.: Niching Methods for Genetic Algorithms. PhD thesis, University of Illinois at Urbana Champaign (1995)
13. Miller, B., Shaw., M.: Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization. In: ICEC 1996 Proceedings, pp. 786–791. IEEE Press, New York (1996)
14. Goldberg, D.E., Wang, L.: Adaptive Niching via Coevolutionary Sharing. In: Genetic Algorithms and Evolution Strategy in Engineering and Computer Science, pp. 21–38. John Wiley & Sons Ltd, West Sussex (1997)
15. Oosten, M., van der Goes, V.: Niching in Evolution Strategies, Technical Report, University of Leiden (2004)

16. Shir, O.M., Bäck, T.: Dynamic Niching in Evolution Strategies with Covariance Matrix Adaptation. In: CEC 2005 Proceedings, pp. 2584–2591. IEEE, Piscataway (2005)
17. Shir, O.M., Bäck, T.: Niche Radius Adaptation in the CMA-ES Niching Algorithm. In: PPSN 2006 Proceedings, pp. 142–151. Springer, Heidelberg (2006)
18. Shir, O.M., Emmerich, M., Bäck, T.: Self-Adaptive Niching CMA-ES with Mahalanobis Metric. In: CEC 2007 Proceedings, pp. 820–827. IEEE Press, Singapore (2007)
19. Törn, A., Zilinskas, A.: Global Optimization, vol. 350. Springer, Heidelberg (1987)

Appendix: Data Associated with *M5*

$$\boldsymbol{T} = \begin{pmatrix} 3 & 1 & 4 & 1 \\ 5 & 9 & 2 & 6 \\ 5 & 3 & 5 & 8 \\ 9 & 7 & 9 & 3 \\ 2 & 3 & 8 & 4 \\ 6 & 2 & 6 & 4 \\ 3 & 3 & 8 & 3 \\ 2 & 7 & 9 & 5 \\ 0 & 2 & 8 & 8 \\ 4 & 1 & 9 & 7 \end{pmatrix}$$

$$\boldsymbol{c} = (1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9)$$

Adaptive Encoding: How to Render Search Coordinate System Invariant

Nikolaus Hansen

Microsoft Research–INRIA Joint Centre, 28 rue Jean Rostand, 91893 Orsay Cedex, France
nikolaus.hansen@inria.fr

Abstract. This paper describes a method for rendering search coordinate system independent, *Adaptive Encoding*. Adaptive Encoding is applicable to any iterative search algorithm and employs incremental changes of the representation of solutions. One attractive way to change the representation in the continuous domain is derived from Covariance Matrix Adaptation (CMA). In this case, adaptive encoding recovers the CMA evolution strategy, when applied to an evolution strategy with cumulative step-size control. Consequently, adaptive encoding provides the means to apply CMA-like representation changes to any search algorithm in the continuous domain. Experimental results confirm the expectation that CMA-based adaptive encoding will generally speed up a typical evolutionary algorithm on non-separable, ill-conditioned problems by orders of magnitude.

1 Introduction

In optimization or search, the problem encoding, that is the choice of the representation of the optimization problem is of utmost importance. A good representation, if available, can render any search problem trivial—finding a proper representation means essentially solving the problem. In an iterative search procedure, in principle, a good problem representation can be iteratively approached, just as a good solution to the problem is approached in the iteration sequence. Indeed, variable metric methods like quasi-Newton methods [2], covariance matrix adaptation (CMA) [7], or estimation of distribution algorithms [8] *implicitly* conduct a representational change. In case of an additive modification of solutions, as for example a mutation in an evolutionary algorithm, a linear change of representation is equivalent with an appropriate linear transformation of the additive mutation [3]. Linear transformations of additive mutation operators, parameterized in step-sizes or covariance matrices, are well studied in evolutionary algorithms [1,8].

In this paper, we sketch an *explicit* framework for an iterative incremental representation change, denoted as *adaptive encoding*. The framework by itself is just about trivial. While the framework is very general, this paper considers subsequently only *linear* changes of the representation *in the continuous domain*.

Searching for a linear representational change in the continuous domain complements the original n-dimensional search problem with a second search problem of size n^2. The advantage from adaptive encoding is that these two search problems are decoupled. Consequently, an effective adaptation of the representation (which can dramatically improve the algorithms performance) can be applied to any underlying search procedure.

In the next section we give the preliminary notations and definitions. In Section 3 the general idea of adaptive encoding is introduced. Section 4 proposes an update rule for the representation, $\mathrm{AE_{CMA}}$, derived from CMA. We can prove that $\mathrm{AE_{CMA}}$ applied to CSA-ES recovers CMA-ES. Section 5 conducts another experimental proof of concept and Section 6 gives a summary and conclusions.

2 Preliminary Notations and Definitions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be an objective function to be minimized. Let a (baseline) search algorithm propose new candidate solutions, $\boldsymbol{x}$, in an iterated procedure and typically evaluate them on f. Let further denote

S the **state space** of the search algorithm;
$\mathcal{A} : S \to S$ an iteration step of the search algorithm $\mathcal{A}$;
$T_B : S \to S$ an invertible transformation, the **decoding** of the state space, the change of representation. The T_B is parameterized by a matrix $\boldsymbol{B}$ and therefore uniquely depends on $\boldsymbol{B}$;
$\boldsymbol{B} \in \mathbb{R}^{n\times n}$ a full rank matrix, representing (i) a new **coordinate system** and a coordinate system transformation in $\mathbb{R}^n$, and (ii) a problem *representation* and linear *decoding* of candidate solutions $\boldsymbol{B} : \boldsymbol{x} \mapsto \boldsymbol{B}\boldsymbol{x}$;
$\mathcal{U} : \mathbb{R}^{n\times n} \times S \to \mathbb{R}^{n\times n}, (\boldsymbol{B}, s) \mapsto \mathcal{U}(\boldsymbol{B}, s)$ the change of representation by updating the matrix $\boldsymbol{B}$. For convenience, we assume that all necessary information to update $\boldsymbol{B}$ is included in the algorithm state s and we may write $\mathcal{U}(\boldsymbol{B})$ instead of $\mathcal{U}(\boldsymbol{B}, s)$;

From these definitions we first remark, that an iteration step of an algorithm can be surrounded by an encoding-decoding step according to

$$\mathcal{A}_B \equiv T_B \circ \mathcal{A} \circ T_B^{-1} \quad , \tag{1}$$

defining algorithm $\mathcal{A}_B : S \to S$, $s \mapsto T_B(\mathcal{A}(T_B^{-1}(s)))$. If T_B is the identity we have $\mathcal{A}_B \equiv \mathcal{A}$.

By definition, *decoded* solutions (phenotypes) are represented in the *given* coordinate system, where also f is evaluated. Accordingly, the algorithm operates, by definition, on *encoded* solutions (genotypes). We usually assume, for convenience and w.l.o.g., that recently evaluated solutions are part of the algorithms state.

Remark 1 (Evaluation of solutions). In order to make use of Eq. (1), we have to ensure that candidate solutions are utilized in their original representation. The solutions must be decoded for evaluation. In other words, $\mathcal{A}$ in Eq. 1 operates on $f \circ \boldsymbol{B}$.

Considering Remark 1, we can execute the algorithm $\mathcal{A}$ in any coordinate system of our choice. The new coordinate system, where the operations of $\mathcal{A}$ are effectively conducted, is defined by $\boldsymbol{B}$. Optimizing $f \circ \boldsymbol{B}$ instead of f already renders $\mathcal{A}$ independent of the given coordinate system (if $\boldsymbol{B}$ is chosen independent of the given coordinate system). Eq. (1) becomes meaningful when we also adapt $\boldsymbol{B}$. We shall choose T_B such that changes of $\boldsymbol{B}$ do not change solutions after they are decoded to their original representation (phenotype).

Finally, we assume to have a performance measure when running an algorithm on an objective function f. The performance measure determines whether one algorithm is better than another. For example, a typical, quantitatively useful measure is the number of candidate solutions evaluated on f until a target function value is achieved.

3 Adaptive Encoding

Equation (1) represents an iteration step of a search algorithm with an additional encoding-decoding procedure. The encoding is, throughout this paper, parameterized by a $n \times n$-matrix; it therefore adds n^2 degrees of freedom. Obviously, the idea is to find a good encoding for algorithm $\mathcal{A}$.

Aim 1 (static encoding). *The goal of finding a good encoding is to find a transformation T_B, such that*

$$T_B \circ \mathcal{A} \circ T_B^{-1} \textit{ outperforms } \mathcal{A} \textit{ on } f$$

The static encoding is usually part of the design of the objective function. Equivalently, the algorithm can be modified specifically in regard to the given objective function (the encoding-decoding can certainly be interpreted as part of the algorithm). The formalism of Aim 1 is not very interesting. To get a more interesting situation, we need to consider an *update* or *adaptation* of the encoding T_B.

Definition 1. *(Adaptive Encoding) Given an algorithm, $\mathcal{A}$, an encoding, T_B, and an update, $\mathcal{U}$, the iteration step of an adaptively encoded algorithm in state $s \in S$ is defined as*

$$s \leftarrow T_B \circ \mathcal{A} \circ T_B^{-1}(s) \tag{2}$$
$$\boldsymbol{B} \leftarrow \mathcal{U}(\boldsymbol{B}, s) \tag{3}$$

where $\leftarrow$ denotes the assignment operator and $T_B \circ \mathcal{A} \circ T_B^{-1}(s) = T_B(\mathcal{A}(T_B^{-1}(s)))$. We write $T_B \circ \mathcal{A} \circ T_B^{-1}\, ; \mathcal{U}(T_B)$ to denote the iteration step of Equations (2) *and* (3).

Obviously, any iterative algorithm $\mathcal{A}$ can be plugged into the adaptive encoding mechanism.

Proposition 1. *(Adaptive Encoding is universal) The Adaptive Encoding from Definition 1 can be applied to any search algorithm.*

Proof. The proposition follows directly from the definition of T_B as invertible mapping from S to S.

Even though Proposition 1 is just about trivial, it is of utmost importance for the implications of our results, because it establishes the general applicability of any effective adaptive encoding.

Analogous to Aim 1, we consider the merits of an adaptive encoding.

Aim 2 (adaptive encoding). *Find an update $\mathcal{U}$, such that for a given T_0 and a given (initial) T_B.*

$$T_B \circ \mathcal{A} \circ T_B^{-1}\,;\, \mathcal{U}(T_B) \textit{ outperforms } T_0 \circ \mathcal{A} \circ T_0^{-1} \textit{ on } f.$$

The left iteration step updates the encoding, the right iteration step applies a constant encoding, T_0, to algorithm $\mathcal{A}$.

Taking only a single iteration step, Aim 2 does not depend on the update $\mathcal{U}$ and it reduces to Aim 1. Consequently, Aim 2 becomes only interesting, when an iteration *sequence* is considered. Indeed, in a realistic automated scenario, Aim 2 can only be achieved in the iteration sequence.

Finally, we define two cases/scenarios when considering Aim 2.

Scenario 1. *(Standard scenario) The initial T_B equals to T_0. Aim 2 shall be satisfied for most given T_0.*

Scenario 2. *(Ambitious scenario) The initial T_B equals to T_0. Aim 2 shall be satisfied* for all *given T_0.*

Satisfying the ambitious scenario implies that no fixed optimal encoding T_B exists and a changing encoding can, in principle, be better than any fixed encoding. Both, the standard and the ambitious scenario are reasonable objectives, depending on the given objective function.

The remainder of this paper proposes and investigates an effective way to implement adaptive encoding as given in Definition 1.

4 Adaptive Encoding Based on Covariance Matrix Adaptation

In order to define an adaptive encoding, we need to specify the encoding of the algorithms state space, $T_B : S \to S$, and the update of the encoding, $\mathcal{U}$. In this section, our aim is to obtain an efficient update $\mathcal{U}$, leaving the choice of T_B as only remaining, algorithm specific design issue. The update is derived from the equations for the covariance matrix update in the $(\mu/\mu_\mathrm{w}, \lambda)$-CMA-ES [4,7], denoted as AE$_\mathrm{CMA}$ in the following, and explicated in Algorithm 1 AE$_\mathrm{CMA}$-Update.

The parameters of AE$_\mathrm{CMA}$-Update are chosen to $\alpha_0 = \frac{\sqrt{n}}{\|\boldsymbol{B}^{-1}(\boldsymbol{m}-\boldsymbol{m}^-)\|}$, with $l_i = \|\boldsymbol{B}^{-1}(\boldsymbol{x}_i - \boldsymbol{m}^-)\|$ we have $\alpha_i = \sqrt{n}\,\max\Big(\frac{l_i}{\beta}, \operatorname*{median}_{j=1,\ldots,\mu}(l_j)\Big)^{-1}$, for $i = 1, \ldots, \mu$ and $\beta = 2$, $\alpha_\mathrm{p} = 1$, $c_\mathrm{p} = \frac{1}{\sqrt{n}}$, $w_i = \frac{\ln(\mu+1)-\ln i}{\mu \ln(\mu+1) - \sum_{j=1}^{\mu} \ln j}$, for $i = 1, \ldots, \mu$, and μ is half of the overall generated number of solutions per iteration (before selection), $c_1 = \alpha_\mathrm{c} \frac{0.2}{(n+1.3)^2+\mu_\mathrm{w}}$, $c_\mu = \alpha_\mathrm{c} \frac{0.2\left(\mu_\mathrm{w}-2+\frac{1}{\mu_\mathrm{w}}\right)}{(n+2)^2+0.2\mu_\mathrm{w}}$ with $\mu_\mathrm{w} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$. Finally $\alpha_\mathrm{c} \approx 1$ must be chosen positive and such that $c_1 + c_\mu \leq 1$. Too large values for α_c potentially lead to a failure. Too small values slow down the adaptation. In any case, a parameter study for α_c is recommended, as conducted below. All parameters are detailed in [5].

The state variables are $\boldsymbol{m}$, $\boldsymbol{p}$ and $\boldsymbol{C}$. The mean $\boldsymbol{m}$ is initialized to the initial solution mean of the search algorithm to which AE$_\mathrm{CMA}$-Update is applied, and initially $\boldsymbol{p} = \boldsymbol{0}$ and $\boldsymbol{C} = \boldsymbol{I}$.

Proposition 2. *Let σ denote a step-size and $\mu_\mathrm{w}^{-1} = \sum_{i=1}^{\mu} w_i^2$. Let $\alpha_p = 1$, $\alpha_0 = \frac{\sqrt{\mu_\mathrm{w}}}{\sigma}$ and $\alpha_i = \sigma^{-1}$, for $i = 1, \ldots, \mu$. Then, the procedure AE$_{CMA}$-Update implements*

Algorithm 1. AE$_{\text{CMA}}$-Update($\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu\}$)
updates the encoding matrix $\boldsymbol{B}$ using the μ recent best-ranked candidate solutions

```
1  given parameters $w_i, c_\text{p}, c_1, c_\mu$, see text
2  given $\boldsymbol{m} \in \mathbb{R}^n$, $\boldsymbol{p} \in \mathbb{R}^n$ and $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ from last iteration
3  let matrices $\boldsymbol{B}^\circ$ orthogonal, and $\boldsymbol{D}$ diagonal, with diagonal elements sorted in ascending
   order , "$\leftarrow$" assigns accordingly
4  $\boldsymbol{m}^- = \boldsymbol{m}$
5  $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_i$                                  // Eq.(3) in [4]
6  set scalars $\alpha_i \geq 0$, for $i = 0, \ldots, \mu$, see text
7  $\boldsymbol{p} \leftarrow (1 - c_\text{p})\boldsymbol{p} + \sqrt{c_\text{p}(2 - c_\text{p})}\,\alpha_0(\boldsymbol{m} - \boldsymbol{m}^-)$                 // Eq.(17) in [4]
8  $\boldsymbol{C}_\mu = \sum_{i=1}^{\mu} w_i\,\alpha_i^2\,(\boldsymbol{x}_i - \boldsymbol{m}^-)(\boldsymbol{x}_i - \boldsymbol{m}^-)^\text{T}$                // rank-$\mu$ matrix
9  set scalar $\alpha_\text{p} \geq 0$, see text
10 $\boldsymbol{C} \leftarrow (1 - c_1 - c_\mu)\,\boldsymbol{C} + c_1\alpha_\text{p}\,\boldsymbol{p}\boldsymbol{p}^\text{T} + c_\mu\boldsymbol{C}_\mu$                 // Eq.(22) in [4]
11 $\boldsymbol{B}^\circ\boldsymbol{D}\boldsymbol{D}\boldsymbol{B}^\circ \leftarrow \boldsymbol{C}$                              // eigendecomposition
12 optionally normalize $\boldsymbol{D}$
13 $\boldsymbol{B} \leftarrow \boldsymbol{B}^\circ\boldsymbol{D}$                                     // encoding matrix
```

the update equations for the evolution path, $\mathbf{p}$*, and the covariance matrix,* $\boldsymbol{C}$*, in the* ($\mu/\mu_\text{W}, \lambda$)*-CMA-ES.*

Proof. Assuming that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\mu$ are the μ best solutions in the recent iteration step, line 5 computes $\boldsymbol{m}$ according to Eq. (3) in [4]. Lines 7 and 10 of AE$_{\text{CMA}}$-Update replicate the covariance matrix update equations (17) and (22) in [4] with added or renamed normalization coefficients, denoted with α. Substituting the coefficients as given above results in the original equations.

The AE$_{\text{CMA}}$-Update implements the covariance matrix update of CMA-ES with additional coefficients α to be specified. Within CMA-ES, this update was designed to operate reliably for any choice of μ [4].

Depending on the application of AE$_{\text{CMA}}$-Update, a slow change of $\boldsymbol{B}$ might be desirable. While $\boldsymbol{C}$ will only change slowly, as long as c_1 and c_μ are small, the decomposition of $\boldsymbol{C}$ does not ensure a similar behavior for $\boldsymbol{B}^\circ$ and $\boldsymbol{D}$. For this reason, the diagonal elements in $\boldsymbol{D}$ are sorted. As an approximation, it might even be sufficient to only decode the solutions for the function evaluation and completely abandon the encoding-decoding of the algorithms state.

AE$_{CMA}$ Recovers CMA-ES. We apply AE$_{\text{CMA}}$ (Algorithm 1) to an evolution strategy with cumulative step-size adaptation (CSA, sometimes also denoted as *path length control*). The AE$_{\text{CMA}}$-($\mu/\mu_\text{W}, \lambda$)-CSA-ES is given in Algorithm 2, where the begin-end block marks the original ($\mu/\mu_\text{W}, \lambda$)-CSA-ES. The following invertible encoding for the state variables in CSA-ES is used.

$$T_B : (\boldsymbol{m}, \boldsymbol{p}_\sigma, \sigma) \mapsto (\boldsymbol{B}\boldsymbol{m}, \boldsymbol{B}^\circ\boldsymbol{p}_\sigma, \sigma) \tag{4}$$

Additionally, the μ best solutions are used as input to AE$_{\text{CMA}}$-Update (line 16 in Algorithm 2). The encoding T_B solely depends on $\boldsymbol{B}$, as $\boldsymbol{B}^\circ$ can be computed from $\boldsymbol{B}$ by normalizing its columns to one. Applying AE$_{\text{CMA}}$-Update to CSA-ES we find.

Algorithm 2. AE$_{\text{CMA}}$-CSA-ES
$\mathcal{N}(\mathbf{0}, \boldsymbol{I}) \in \mathbb{R}^n$ indicates a $(0,1)$-normal distribution in each coordinate
$i(f)$ indicates the index of the i-th best solution, e.g., $\boldsymbol{x}_{1(f)} = \underset{i=1,\ldots,\lambda}{\arg\min}\,\{f(\boldsymbol{x}_i)\}$
Shaded areas implement the adaptive encoding, AE$_{\text{CMA}}$, also updating $\boldsymbol{B}$ and $\boldsymbol{B}^\circ$
The begin-end block embraces the original CSA-ES minimizing $f \circ \boldsymbol{B}$

```
1  initialize $\boldsymbol{m} \in \mathbb{R}^n$ (distribution mean), $\boldsymbol{p}_\sigma = \mathbf{0}$ (evolution path), $\sigma > 0$ (step-size)
2  initialize $\boldsymbol{B} = \boldsymbol{B}^\circ = \boldsymbol{I}$ (encoding matrices)
3  repeat
4      $\boldsymbol{m} \leftarrow \boldsymbol{B}^{-1}\boldsymbol{m}$
5      $\boldsymbol{p}_\sigma \leftarrow \boldsymbol{B}^{\circ\mathrm{T}}\boldsymbol{p}_\sigma$
6      begin
7          $\boldsymbol{x}_i = \boldsymbol{m} + \sigma\,\mathcal{N}_i\,(\mathbf{0}, \boldsymbol{I})\,,$   for $i = 1, \ldots, \lambda$
8          $f_i = f \circ \boldsymbol{B}\,(\boldsymbol{x}_i) = f(\boldsymbol{B}\,\boldsymbol{x}_i)\,,$ for $i = 1, \ldots, \lambda$   // decode to evaluate
9          $\boldsymbol{m}^- = \boldsymbol{m}$
10         $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\,\boldsymbol{x}_{i(f)}$
11         $\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\,\boldsymbol{p}_\sigma + \sqrt{c_\sigma\,(2 - c_\sigma)\mu_{\mathrm{w}}}\,\frac{1}{\sigma}(\boldsymbol{m} - \boldsymbol{m}^-)$
12         $\sigma \leftarrow \sigma\,\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{E\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|} - 1\right)\right)$
13     end
14     $\boldsymbol{m} \leftarrow \boldsymbol{B}\boldsymbol{m}$
15     $\boldsymbol{p}_\sigma \leftarrow \boldsymbol{B}^\circ\boldsymbol{p}_\sigma$
16     $\boldsymbol{B}, \boldsymbol{B}^\circ \leftarrow$ AE$_{\text{CMA}}$-Update$(\{\boldsymbol{B}\boldsymbol{x}_1, \ldots, \boldsymbol{B}\boldsymbol{x}_\mu\})$   // update $\boldsymbol{B}$ and $\boldsymbol{B}^\circ$
17 until stopping criterion is met
```

Theorem 1 (Recovery of CMA-ES). *Let T_B given in Eq.* (4) *and the scalars for AE$_{CMA}$-Update in each iteration given in Proposition 2, then the AE$_{CMA}$-$(\mu/\mu_{\mathrm{w}}, \lambda)$-CSA-ES (Algorithm 2) implements the $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES.*

Proof. Due to the space limitations, the proof is provided in [5].

Theorem 1 supports the hypothesis that AE$_{\text{CMA}}$-Update is an effective way to update the representation matrix $\boldsymbol{B}$ in evolutionary search algorithms, as CMA-ES efficiently adapts the principle axes of the coordinate system, where the independent sampling takes place. In the next section, another application of AE$_{\text{CMA}}$-Update is realized.

5 Yet Another Experimental Proof of Concept

While AE$_{\text{CMA}}$-Update has proved to be effective with CSA-ES [7], in this section we provide another case study. To underline the general applicability of AE$_{\text{CMA}}$-Update we consider a baseline algorithm that (i) exploits the given coordinate system, and (ii) generates distributions that are rather different from Gaussians, providing a test scenario that is rather different from CSA-ES. We use a simple but functional algorithm that utilizes a Cauchy distribution in a derandomized adaptation framework, denoted as $(1, \lambda)$-Cauchy-ES, with $\lambda = 10$ (see the begin-end block in Algorithm 3). Despite this choice

Algorithm 3. AE$_{\mathrm{CMA}}$-$(1,\lambda)$-Cauchy-ES
The begin-end block embraces the $(1,\lambda)$-Cauchy-ES minimizing $f\circ\boldsymbol{B}$
Shaded areas implement the adaptive encoding
$\boldsymbol{R}_i\in\mathbb{R}^n$ is standard Cauchy distributed in each component
$1(f)$ indicates the index of the best solution $\boldsymbol{x}_{1(f)}=\arg\min_{i=1,\dots,\lambda}\{f(\boldsymbol{x}_i)\}$

```
1  initialize $\boldsymbol{x}\in\mathbb{R}^n$ and diagonal matrix $\sigma$ (step-size matrix)
2  initialize $\boldsymbol{B}=\boldsymbol{I}$ (encoding matrix)
3  repeat
4      $\boldsymbol{x}\leftarrow\boldsymbol{B}^{-1}\boldsymbol{x}$
5      begin
6          $\boldsymbol{x}_i=\boldsymbol{x}+\sigma\boldsymbol{R}_i$,    for $i=1,\dots,\lambda$
7          $f_i=f\circ\boldsymbol{B}(\boldsymbol{x}_i)=f(\boldsymbol{B}\boldsymbol{x}_i)$,  for $i=1,\dots,\lambda$    // decode to evaluate
8          $\boldsymbol{x}\leftarrow\boldsymbol{x}_{1(f)}$
9          $\sigma_{jj}\leftarrow\sigma_{jj}\exp\left(\frac{1}{2n}\left(\frac{1}{2}\mathrm{sign}(|R_{1(f),j}|-0.9)+\mathrm{sign}\sum_{i=1}^{n}\mathrm{sign}(|R_{1(f),i}|-1)\right)\right)$
10     end
11     $\boldsymbol{x}\leftarrow\boldsymbol{B}\boldsymbol{x}$
12     $\boldsymbol{B}\leftarrow$ AE$_{\mathrm{CMA}}$-Update$(\{\boldsymbol{B}\boldsymbol{x}_1,\dots,\boldsymbol{B}\boldsymbol{x}_\mu\})$    // update $\boldsymbol{B}$
13 until stopping criterion is met
```

(lead by simplicity and personal preference), neither comma-selection (non-elitism) nor derandomization nor a small population size are fundamental prerequisites for applying AE$_{\mathrm{CMA}}$. The $(1,\lambda)$-Cauchy-ES samples new solutions without dependencies between variables in the given coordinate system, because σ is a diagonal matrix. Rendering the $(1,\lambda)$-Cauchy-ES coordinate system independent results in correlations between variables (even if $\sigma=\boldsymbol{I}$), because the Cauchy distribution is highly anisotropic [6]. The invertible encoding

$$T_B:(\boldsymbol{x},\sigma)\mapsto(\boldsymbol{B}\boldsymbol{x},\sigma)\tag{5}$$

is used, where the step-size matrix σ is not transformed. An appropriate mapping for a covariance matrix $\sigma^2\mapsto\boldsymbol{B}\sigma^2\boldsymbol{B}^{\mathrm{T}}$ would not preserve the diagonal property, while arguably $T_B^{-1}:\boldsymbol{C}\mapsto\mathtt{diag}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{B})$ could be used. In contrast, using $\boldsymbol{B}\mathtt{diag}(\sigma)$ as mapping for only the diagonal of σ cannot be recommended, because very small diagonal entries can occur accidentally. Because σ is not encoded, it is important that changes of $\boldsymbol{B}$ remain modest.

Using $\boldsymbol{B}^\circ$ instead of $\boldsymbol{B}$ in Eq. (5) is a possible alternative and investigated below. In this case, the step-size matrix σ needs to learn the scaling that can be otherwise provided by the diagonal matrix $\boldsymbol{D}$ in Algorithm 1.

Test functions. Testing on a number of functions, we always found the expected effect from AE$_{\mathrm{CMA}}$. Exemplarily, we show simulations on two quadratic test functions, falling into Scenario 1 from Section 3.

$$f_{\mathrm{elli}}(\boldsymbol{x})=\sum_{i=1}^{n}10^{6\,\frac{i-1}{n-1}}y_i^2\quad\text{and}\quad f_{\mathrm{cigtab}}(\boldsymbol{x})=y_1^2+10^4\sum_{i=2}^{n-1}y_i^2+10^8y_n^2\ ,\tag{6}$$

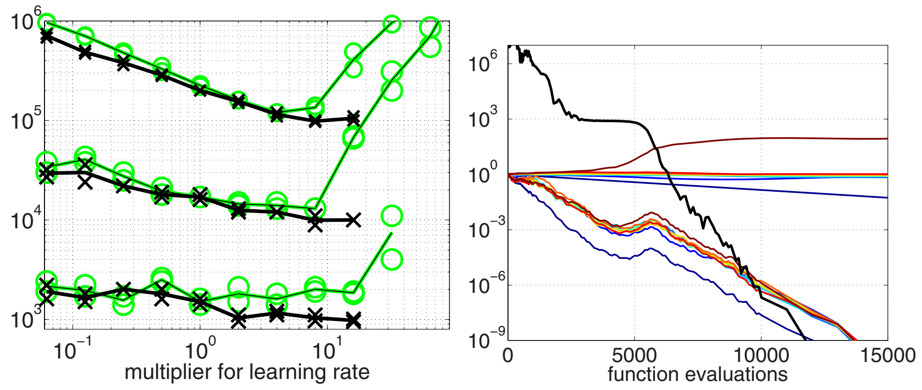


Fig. 1. Simulation of the AE$_{\rm CMA}$-$(1, \lambda)$-Cauchy-ES. Left: number of function evaluations to reach function value 10^{-9} on the rotated $f_{\rm elli}$ versus the multiplier $\alpha_{\rm c}$ for the learning rate of $\boldsymbol{B}$ in 3-, 10- and 30-D (from bottom to top). Symbols $\times = \boldsymbol{B}$ and $\bigcirc = \boldsymbol{B}^\circ$. For each set-up two trials were conducted. Right: time evolution on the rotated $f_{\rm cigtab}$ in 10-D, shown are the objective function value (bold single graph), diagonal elements of the step-size matrix σ (lower group of curves) and diagonal elements of $\boldsymbol{D}$ in Algorithm 1 (smooth upper graphs).

where $\boldsymbol{y} := \boldsymbol{O}\boldsymbol{x}$ and $\boldsymbol{O} = [\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n]$ implements an angle-preserving, linear transformation, *i.e.* $\boldsymbol{O}$ is orthogonal. The basis $\boldsymbol{O}$ was either chosen as identity $\boldsymbol{I}$ (*axis-parallel* case), or each column was sampled uniformly distributed on the unit hypersphere, orthogonalized to the previous columns and normalized to one (*rotated* case). For each trial a new basis was sampled. Further initial values were $\boldsymbol{x} = (1, \ldots, 1)^{\rm T}$ and $\sigma = \boldsymbol{I}$. In the following, if a single trial is shown, it represents a typical trial.

Choosing parameters for AE$_{CMA}$. For applying AE$_{\rm CMA}$ to the $(1, \lambda)$-Cauchy-ES, we conduct a minimalistic parameter study for the multiplier, $\alpha_{\rm c}$, of the learning rate for the matrix $\boldsymbol{B}$. Further parameters for the AE$_{\rm CMA}$-Update follow the settings from Section 4, accordingly, we use $\mu = \lambda/2 = 5$. We test two cases, (i) using Eq. (5) and (ii) replacing $\boldsymbol{B}$ with $\boldsymbol{B}^\circ$ in Eq. (5). The remaining set-up is minimalistic. We test on the rotated $f_{\rm elli}$ in 3-, 10- and 30-D, vary $\alpha_{\rm c}$ by factors of 2 and $1/2$ and measure the number of function evaluations to reach function value 10^{-9}, twice for each set-up. Results are shown in Figure 1, left. Missing points for large values of $\alpha_{\rm c}$ (to the right) indicate that at least one run did not succeed. Large values lead to a failure, because the condition number of matrix $\boldsymbol{D}$ (line 11 in Algorithm 1) diverges. Using $\boldsymbol{B}^\circ$ is less prone to a failure. When reducing $\alpha_{\rm c}$ to small values, the number of function evaluations will increase at most linearly with ${\alpha_{\rm c}}^{-1}$.

Only for large values of $\alpha_{\rm c}$ the performance is remarkably different for $\boldsymbol{B}$ and $\boldsymbol{B}^\circ$. With increasing $\alpha_{\rm c}$, first $\boldsymbol{B}^\circ$ becomes worse, but finally $\boldsymbol{B}$ fails earlier than $\boldsymbol{B}^\circ$. Nevertheless, the designated default value $\alpha_{\rm c} = 1$ is applicable in both cases: the value is more than ten times larger than a value that leads to a failure and the performance loss to the best setting, does not exceed a factor of two. We retain using $\boldsymbol{B}$ as in Eq. (5) in the following.

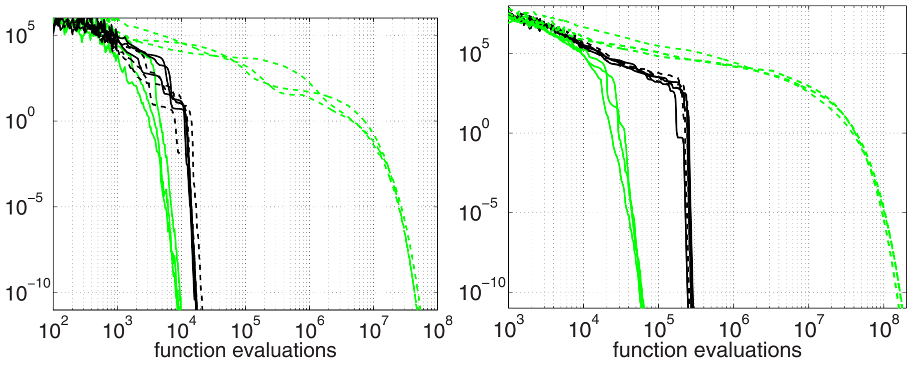


Fig. 2. Simulation of AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES (bold) and $(1, \lambda)$-Cauchy-ES (light) on the axis-parallel (solid) and the rotated (dashed) f_{elli} in 10-D (left) and 30-D (right). Shown is the objective function value of respectively 3 trials over time. In the rotated case, the AE$_{\text{CMA}}$ improves the performance by a factor of roughly one thousand.

Figure 1 (right) shows a single run on the rotated f_{cigtab} in 10-D. The function topology is successfully adapted as the optimum is approached quickly after about 5000 function evaluations. The single large dispersion value in the distribution, relating to y_1 in Eq. (6), is mainly represented in the matrix $\boldsymbol{D}$ (upper graph), while the single small value, relating to y_n, is mainly represented in σ, in particular in the early stage.

The comparison. Completing the picture, we compare the AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES with the $(1, \lambda)$-Cauchy-ES on f_{elli}. In Figure2, three runs are shown for each algorithm on the axis-parallel and on the rotated function in 10-D (left) and 30-D (right).

The performance of the AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES is virtually independent of rotation (only the initialization is still different in both cases). On the axis-parallel function, AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES becomes about two to ten times slower than $(1, \lambda)$-Cauchy-ES (for reaching function value 10^{-10} in 10-D and function value 100 in 30-D respectively). On the rotated function, AE$_{\text{CMA}}$-$(1, \lambda)$-Cauchy-ES becomes between 200 and 2000 times(!) faster (for reaching function value 10^{-1} in 30-D and function value 10^{-10} in 10-D respectively). The application of the AE$_{\text{CMA}}$ was apparently successful (the CMA-ES is still roughly four times faster in this particular case). The trade-off when AE$_{\text{CMA}}$ is applied with the axis-parallel function is comparatively small, but increases with increasing dimension. By fixing the transformation $\boldsymbol{B}$ for some time in the beginning of the optimization, this trade-off can be eliminated.

6 Summary and Conclusions

We have outlined an *adaptive* change of representation in iterative search, denoted as *adaptive encoding* (AE): after each iteration step, (i) the algorithms state is "decoded", (ii) the encoding mechanism is adapted, and (iii) the algorithms state is "encoded" again for the next iteration. Additionally, candidate solutions are decoded for their evaluation on the objective function. In the continuous search domain, practical implications from this simple procedure are surprisingly far-reaching. The sophisticated update of the covariance matrix in the CMA-ES can be entirely formulated as a change of representation of an encoding matrix $\boldsymbol{B}$ (Proposition 2). Applying this representation change, $\mathrm{AE}_{\mathrm{CMA}}$, to an evolution strategy with cumulative step-size control, the CMA-ES is recovered (Theorem 1)—proving that an effective representation change can be entirely decoupled from the underlying search algorithm. Addressing an important open problem in evolutionary computation [9], the representation change implicitly induced by the covariance matrix adaptation in the CMA-ES becomes available for any continuous domain search algorithm—$\mathrm{AE}_{\mathrm{CMA}}$ *can render any search algorithm independent of the coordinate system*, in particular rotationally invariant.

We conjecture that on various non-separable ill-conditioned problems $\mathrm{AE}_{\mathrm{CMA}}$ will typically speed up population-based search methods by orders of magnitude. A case study of $\mathrm{AE}_{\mathrm{CMA}}$ supports our conjecture: the baseline algorithm has become roughly thousand times faster on the non-separable problems. While the principle of adaptive encoding is quite general, we anticipate successful applications (e.g. using Algorithm 1) in particular for population-based, stochastic search algorithms in the continuous domain.

References

1. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
2. Davidon, W.: Variable Metric Method for Minimization. SIAM J. Optim. 1, 1 (1991)
3. Hansen, N.: Invariance, self-adaptation and correlated mutations in evolution strategies. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 355–364. Springer, Heidelberg (2000)
4. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a new evolutionary computation. Advances on estimation of distribution algorithms, pp. 75–102. Springer, Heidelberg (2006)
5. Hansen, N.: Adaptive encoding for optimization. Research Report 6518, INRIA (April 2008), `http://hal.inria.fr/inria-00275983/en`
6. Hansen, N., Gemperle, F., Auger, A., Koumoutsakos, P.: When do heavy-tail distributions help? In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 62–71. Springer, Heidelberg (2006)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
8. Larrañaga, P.: A review on estimation of distribution algorithms. In: Larrañaga, P., Lozano, J. (eds.) Estimation of distribution algorithms, pp. 80–90. Kluwer Academic Publishers, Dordrecht (2002)
9. Salomon, R.: Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. BioSystems 39(3), 263–278 (1996)

Supervised and Evolutionary Learning of Echo State Networks

Fei Jiang[1,2], Hugues Berry[1], and Marc Schoenauer[2]

[1] Alchemy, INRIA Saclay, 28, rue Jean Rostand, 91893 Orsay Cedex, France
[2] TAO, INRIA Saclay & LRI (UMR CNRS 8623), Bât 490, Université Paris-Sud, 91405 Orsay Cedex, France

Abstract. A possible alternative to topology fine-tuning for Neural Network (NN) optimization is to use Echo State Networks (ESNs), recurrent NNs built upon a large reservoir of sparsely randomly connected neurons. The promises of ESNs have been fulfilled for supervised learning tasks, but unsupervised ones, e.g. control problems, require more flexible optimization methods – such as Evolutionary Algorithms. This paper proposes to apply CMA-ES, the state-of-the-art method in evolutionary continuous parameter optimization, to the evolutionary learning of ESN parameters. First, a standard supervised learning problem is used to validate the approach and compare it to the standard one. But the flexibility of Evolutionary optimization allows us to optimize not only the outgoing weights but also, or alternatively, other ESN parameters, sometimes leading to improved results. The classical double pole balancing control problem is then used to demonstrate the feasibility of evolutionary (i.e. reinforcement) learning of ESNs. We show that the evolutionary ESN obtain results that are comparable with those of the best topology-learning methods.

Keywords: Neural networks, Evolutionary algorithms, Control.

1 Introduction

It has long been known to Neural Networks practitioners that a good design for the topology of the network is an essential ingredient for a successful application of Neural Networks to a given learning task. The critical issue then becomes that of learning the appropriate weights. Echo State Networks (ESNs) [12], that were recently proposed for supervised learning of time series, can be seen as an alternative approach based on a large *reservoir* of neurons with random, constant (non-learned) and sparse connectivity. Learning is thus restricted to the outgoing connections only. In the supervised learning case, this efficiently transforms the learning process into a simple quadratic optimization problem. The situation changes dramatically with unsupervised learning tasks, such as control ones: no input-output example being available, the learning problem can no longer be set as quadratic. Evolutionary Computation provides a possible solution for such situations, as long as some fitness is available. This paper addresses the following issues: are Evolutionary Algorithms (EAs) a viable method to train ESNs in general and on reinforcement learning tasks in particular? Furthermore, are ESNs an alternative to topology learning in the framework of control problems? Finally, Evolutionary Algorithms can learn to adjust more than just the weights of the outgoing connections of the ESN. Does this improve the learning power of ESNs?

The paper is organized as follows. Section 2 introduces ESNs and our evolutionary algorithm. The supervised task case (time-series prediction) is addressed in Section 3. Moreover, Evolutionary Learning opens up the field of reinforcement learning to ESNs. We address this issue with a canonical example, the double pole balancing problem [17,10,4], in Section 4. Finally, Section 5 sums up the paper and sketches directions for on-going and further researches.

2 Background

2.1 Echo State Networks

Echo state networks (ESN) are discrete time, continuous state, recurrent neural networks using a sigmoidal activation function for all neurons [12]. A typical ESN is shown in figure 1: the input layer is totally connected to the hidden layer (the *reservoir*) whose neurons are themselves totally connected to the output layer. Note that the output layer can also be connected backward to the reservoir. To generate the reservoir, one connects N neurons randomly (with independent uniform distribution) up to a user-defined connection density α. The weight of these connections are randomly chosen, then scaled so that the spectral radius of the reservoir, ρ (i.e. the largest modulus among the eigenvalues of the reservoir weight matrix) is less than a prescribed value < 1 (see e.g. [13]). The main point in ESN is that only the weights from the reservoir nodes to the output ones are to be learned. Any supervised learning problem using some mean-square error objective thus reduces to a quadratic optimization problem that can be

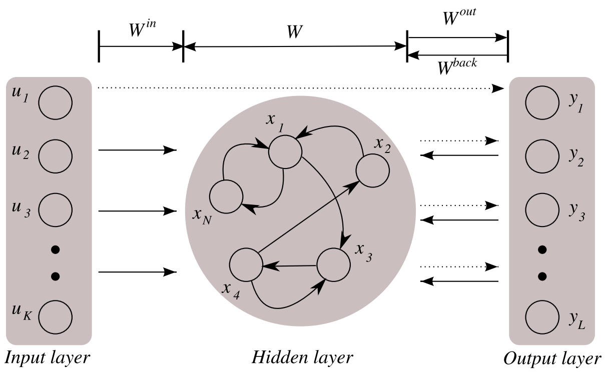


Fig. 1. Schematic view of an Echo State Network. Plain arrows stand for weights that are randomly chosen and remain fixed, while dashed arrows represent the weights to be optimized.

quickly solved by any deterministic optimization procedure, even for very large values of N. ESNs have been shown to perform surprisingly well in the context of supervised learning, in particular for time series prediction. They have also been successfully used in the context of (supervised) robot control learning [14]. The idea beyond Evolutionary Learning for Echo State Networks is to replace the gradient descent used to optimize the outgoing weights in Jaeger's approach [13] by an Evolutionary Algorithm (EA). We first present the Evolutionary Algorithm that will be used throughout the paper, the Covariance Matrix Adaptation Evolution Strategy, aka CMA-ES.

2.2 CMA-ES

The CMA-ES is a well-established and state-of-the-art Evolutionary Algorithm in continuous domain evolutionary computation [8,9,7]. At iteration step t, $\lambda > 1$ offspring individuals $\boldsymbol{x} \in \mathbb{R}^n$ are generated by sampling a multi-variate normal distribution: $\boldsymbol{x} = \boldsymbol{m}^t + \sigma^t \times \mathcal{N}(\mathbf{0}, C^t)$, where $\boldsymbol{m}^t$ is the average of the best individuals of the previous generation, $\mathcal{N}(\mathbf{0}, C^t)$ is a normally distributed variable with mean $\mathbf{0}$ and $n \times n$ covariance matrix C^t, and $\sigma^t > 0$ is a scaling parameter, the step-size. After those λ individuals have been sampled, evaluated on f, and sorted according to their objective function values, the distribution parameters $\boldsymbol{m}^t$, σ^t, and C^t are updated for a new iteration step using the sorted population and cumulated information about the whole optimization path. It has been shown experimentally that the covariance matrix C^t approximates the inverse of the Hessian matrix of the problem at hand near the optimum, and CMA-ES can hence be considered a quasi-second order optimization method. Importantly, CMA-ES is almost a parameter-free algorithm. Only the number of offsprings λ is crucial to the evolution success and must possibly be adapted to account for the ruggedness of the fitness landscape at hand. In our case, the default value [9], that increases logarithmically with the dimension n of the problem (number of unknown parameters): $\lambda = 4 + 3\ln(n)$, was found to be well adapted.

3 Supervised Learning of ESN

In order to validate the Evolutionary approach to ESN learning, we first replicate Jaeger's initial setting [12], but using an Evolutionary Algorithm in lieu of its gradient-based quadratic optimization procedure.

3.1 The Original Settings

In this toy example, the aim is to train the network to produce a univariate time-series output, $y_{teach}(n) = \frac{1}{2}u^7(n)$ (where n is time) from a univariate input given by $u(n) = \sin(n/5)$. The network output is given by $y(n) = f(\sum_{i=1}^{N} w_i^{out} \times x_i(n))$, where w_i^{out} denotes the weight of the i-th output connection, $x_i(n)$ is the state of the i-th neuron, $f(x) = (1 - e^{-ax})/(1 + e^{-ax})$ and a is the half-slope of f at zero activation. Like in Jaeger's original paper, the reservoir consists of $N = 100$ randomly connected neurons (independent uniform distribution). Its weights are set to 0, +0.4 or -0.4 with probabilities 0.95, 0.025, 0.025 respectively (sparse connectivity of 5%). They are then scaled so that the spectral radius of the reservoir is $\rho \approx 0.88$. The input weights (from the input to all neurons in the reservoir) are set to $+1$ or -1 with equal probability. Direct links from inputs to outputs or backward links from outputs to the reservoir are not used here. The fitness to minimize is the Mean Square Error of the network, computed between time steps 101 and 300: $\text{mse}_{train} = 1/200 \sum_{n=101}^{300} (y(n) - \text{atanh}(y_{teach}(n)))^2$.

3.2 Which Parameter to Optimize?

In Jaeger's original paper [12], the output weights were optimized with a gradient method, resulting in a reported error mse $\approx 3.5 \times 10^{-15}$ [12]. But a critical parameter in ESN tuning seems to be the spectral radius, that is usually advised to be < 1 [12] though different values have been proposed in the literature for different problems. Hence it seems a good idea to use the spectral radius as a free parameter to be optimized by CMA-ES: it only adds one dimension to the problem. The procedure goes as follows: the weights of the recurrent connections within the reservoir are first scaled so that the spectral radius of the connection matrix takes the value prescribed by the additional optimized parameter. The weights are of course set back to their original values before the evaluation of next individual. Jaeger's original sigmoidal function was tanh, corresponding to the case $a = 2$ for the transfer function f above. However, if both the output weights and the sigmoid slopes a are optimized, the dimension of the optimization problem is twofold. Hence, we examined the case where only the slopes a are optimized.

3.3 Comparative Measures

Because CMA-ES, like all EAs, is a stochastic optimization procedure, no strong conclusion can be drawn in absence of a thorough statistical analysis of the performances. Here 15 different networks have been used, and for each network, 5 runs of CMA-ES were launched with different random seeds (and hence starting points). To have a global performance measure, we used an estimator of the success performance called the "SP1 measure" [7,1], which is the number of evaluation of the fitness function that is needed to reach a given fitness level, divided by the fraction of runs that did reach that fitness value. SP1 can thus be viewed as the computational effort required to reach a given performance level.

3.4 Results

Three variants of the ESN evolutionary optimization have thus been compared: *(i)* optimizing the output weights only, denoted *Std* in the following; *(ii)* optimizing the output weights *plus* the spectral radius, denoted *Rho*; and *(iii)* optimizing the sigmoidal slopes only, denoted *Slopes*. Figure 2 shows the SP1 plots for a 100 neuron reservoir and confirms that CMA-ES (*Std*) can be as precise as the gradient method reported in [12] (i.e. with a *mse* of the order of 10^{-15}), though undoubtedly requiring a much greater computational effort.

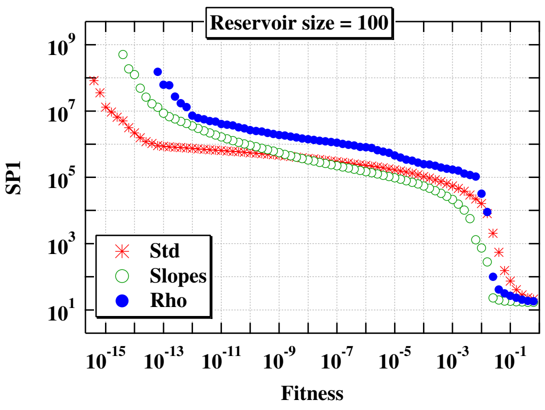


Fig. 2. Comparative SP1 measures for the case $N = 100$, in log-log scale

Interestingly, the results show that optimizing only the reservoir slopes (*Slopes*) yields precisions that are also similar to the original ESN learning method. Note however that with smaller reservoir sizes (e.g. $N = 30$), optimizing the reservoir neuron slopes (*Slopes* variant) yielded even better fitness than the standard procedure (not shown). This, however, has a cost and requires almost 100-folds more evaluations, due to the fact that very few runs do find such low fitness values. Finally, Figure 2 also evidences that increasing the search space fails to improve precision: the *Rho* variant yields the worst precision in this supervised task. Taken together, these results validate the use of Evolutionary Learning for supervised tasks. We now turn to the study of a reinforcement learning task.

4 Reinforcement Learning of ESN

The double pole balancing problem without velocity information is a benchmark learning task for the evaluation of neuroevolution methods - i.e. methods that evolve both the topology and the weights of neural networks [17,6,5,10,4]. Albeit they don't belong to supervised learning methods, evolutionary methods are based on the evaluation of some individual fitness. This fitness can be considered as a feedback or a kind of reward emitted by the environment, so that such neuroevolution methods are considered as reinforcement learning methods. The system consists of a cart (mass =1 kg) moving along the x axis, and two poles of different lengths ($l_1 = 1\,\mathrm{m}$, $l_2 = 0.1\,\mathrm{m}$) and masses ($m_1 = 0.1\,\mathrm{kg}$, $m_2 = 0.01\,\mathrm{kg}$) that are connected to the cart by a hinge. The poles have a single degree of freedom (their angle θ_1 and θ_2 w.r.t. the vertical). The challenge is to keep both poles up (i.e. within given bounds for their angles) as long as possible using the ESN output, which is interpreted as a force F_x applied to the cart ($F_x \in [-10\,\mathrm{N}, 10\,\mathrm{N}]$). In all experiments (in this paper as well as in previous works), the dynamics of this mechanical system was solved using fourth-order Runge-Kutta method with a step size of 0.01 s.

4.1 Fitness(es)

To avoid heavy computational cost many, if not all, previous works in the evolutionary literature addressing the double pole balancing problem [6,5,17,10,4] have used a simplified fitness (thereafter referred to as F_{cheap}): a single trial is run for every individual in the population, starting from the same state ($\theta_1(0) = 4.5^o, \dot{\theta}_1(0) = \theta_2(0) = \dot{\theta}_2(0) = x(0) = \dot{x}(0) = 0$). The simulation stops if one of the poles falls, i.e. the system leaves the success domain $x \in [-2.4\,\text{m}, 2.4\,\text{m}]$ and $\theta_1, \theta_2 \in [-36^o, 36^o]$ (no solution found) or if the poles remain up for $1,000$ time steps (successful individual). The fitness function F_{cheap} is then:

$$F_{cheap} = 10^{-4}t + 0.9 f_{stable}, \text{ with}$$

$$f_{stable} = \begin{cases} 0 & \text{if } t < 100 \\ \dfrac{0.75}{\sum_{i=t-100}^{t} \left(|x(i)| + |\dot{x}(i)| + |\theta_1(i)| + |\dot{\theta}_1(i)|\right)} & \text{otherwise} \end{cases}$$

where t denotes the number of time steps during which the system remained inside the success domain and f_{stable} quantifies the cart stability during the last 100 time steps. At every generation, the best individual for fitness F_{cheap} undergoes two generalization tests. The first test is passed if the individual keeps the system within the success domain during $100,000$ further time steps. The second test is passed if the individual as well succeeds in balancing the system for $1,000$ time steps starting from 625 different initial positions. When one individual succeeds for at least 200 of those 625 trials, the run is stopped and this individual is returned as the solution.

However, though this simplified fitness does save a lot of computational resources, it is a poor fitness with respect to the overall goal of the optimization. For instance, individuals are commonly obtained that have a very high fitness but never pass the first generalization test, while some others pass all generalization tests but with a rather low F_{cheap}. Hence we propose here a new fitness ($F_{gen.}$) that takes into account all 3 tests described above: $F_{gen.} = F_{cheap} + 10^{-5} n_I + 30 n_S/625$ where n_I is the number of iterations where the system was maintained within the success domain during the first generalization test, and n_S is the number of generalization trials passed by the controller during the second one. The constants 10^{-5} and 30 were chosen by trial and error.

4.2 Experimental Conditions

The size of the reservoir was fixed here to $N = 20$: initial experiments indicated that larger reservoirs did not improve the results. To study the variability with respect to the reservoir topology, 20 different reservoirs were generated and 11 independent runs of CMA-ES were made for each reservoir. Each reservoir was initialized as described in section 3.1, except for the fixed weights: here, the reservoir connectivity was 10% and non-zero weights were randomly initialized between $[-1, 1]$. At the beginning of each run, the activity of all neurons in the reservoir was zeroed, and the network was run for 20 iterations before control actually began and the fitness started to accumulate. As mentioned in Section 2.2, CMA-ES is almost a parameter-free algorithm. However, Igel advises in his paper [10] to impose a lower bound on the actual lower eigenvalue of the covariance matrix during CMA-ES runs. Indeed, our preliminary results confirmed that, without this constraint, the solutions systematically evolves toward "Bang-Bang" types of motor control, that do not seem very efficient for the task at hand. We were able to solve this issue by imposing a lower bound of 0.05 on the step-size σ. Finally, the presented results were obtained with the *Std* and *Rho* variants of the evolutionary ESN learning (Section 3.2).

4.3 Results and Discussion

All results are summarized in Table 1. Every line in the table gives the results of one variant of the algorithm (two spectral radii, 0.6 and 0.95 were tried for the *Std* variant, and variant *Std - Opt* will be discussed later). For each variant, the 220 runs (11 runs for each of the 20 different reservoir initializations) are here grouped together. Each sub-table shows the average number of needed evaluations **averaged over the successful runs** (column *Avg Eval.*), its standard deviation (*Std Dev.*), the number of tests (out of 625) passed during the third generalization test (*Generalization*), and, most importantly, the percentage of success (*% success*), i.e. of runs where the best individual did pass the 3 tests. Using the "cheap" fitness F_{cheap}, a first striking result is the very low performance of the *Std* variants (whatever the spectral radius): less than 7% of the runs did pass the 3 generalization tests in these cases. Things are better for the *Rho* variant: more than half of the runs succeeded, with an average cost of $23,571$ evaluations, which amounts to an SP1 value of about $45,300$. This value is still worse than NEAT ($\approx 33,000$ evaluations [17]) and AGE ($\approx 25,000$ evaluations [4]), but within the same order of magnitude.

As expected, the results really improve when using the new fitness, that takes into account the generalization ability of the network: the *Rho* variant almost always find a solution (except for one run out of 220). Even the *Std* ones improve a lot over their results with the cheap fitness. More importantly, using the new fitness allows all variants to reach performances that are comparable to those of NEAT ($\approx 33,000$ evaluations [17]) and AGE ($\approx 25,000$ evaluations [4]), though of course those results can hardly be compared, as they were obtained using a different fitness. Indeed, the found SP1 values for *Std-0.60*, *Std-0.95*, and *Rho* variants respectively were $19,342$, $19,808$ and $21,658$.

Spectral Radius. It has always been advocated by ESN pioneers that the upper bound on the spectral radius was important for successful ESN use, and the results for both *Std* variants with different spectral radius seem to confirm this. However, the most remarkable fact here is that for all settings, the *Rho* variant, that explicitly optimizes the spectral radius, almost always gives the best results. This is surprising when compared to the situation in the supervised context (Section 3.4), where the *Rho* variant performed the worst of all.

Further experiments were run, using the *Std* variant but fixing the spectral radius to the final value found by the *Rho* method (see the lines "*Std – Opt*" in

Table 1. Experimental results for the double pole balancing

	Cheap Fitness				New Fitness			
Method	Avg. Eval.	Std. Dev.	Genera-lization	% success	Avg. Eval.	Std. Dev.	Genera-lization	% success
Std - 0.95	14960	6291	234	6.8%	16303	11511	209	82.3%
Std - 0.60	16639	17037	225	6.8%	16886	11073	211	87.3%
Rho	23571	10175	241	52.7%	19796	6770	224	91.4%
Std - Opt	19168	21782	232	9.5%	15965	11813	208	86.8%

Table 1). Though it generally slightly improves the results over an arbitrary value like 0.6 or 0.95, it does not allow to reach the same level of performance as the *Rho* method itself. The important feature is thus that ρ is allowed to vary during the optimization, and not the final value it reaches. A final advantage about the *Rho* variant, is that it seems to be able to provide controllers that generalize very well, if evolution (using the new fitness) is continued after the first network has passed the 200-tests of the last generalization test: all resulting networks are able to successfully solve more than 500 out of the 625 test cases, with a peak at 555 for one network. Unfortunately, the previously published studies do not report this kind of result, except for one sentence in [4] that mentions that one network successfully solved 525 test cases.

Reservoir topologies. The results obtained with the the double pole problem were found to vary a lot among the different (random) realizations of the connections (i.e. the non-zero weights) in the reservoir, for the same value of the density of connection. Indeed, in the case of methods with low performance, all the successful runs often stem from a small number of initial reservoir topologies, while a majority the initial reservoir topologies fail to generate even a single success. Together with the differences noted in the supervised learning context, this makes a clear picture that the topology of the reservoir matters. Why, and how to take advantage of this fact, is left to further work.

The question is now open: whereas reservoir computing has been proposed as a possible alternative to fine tuning of the weights in Neural Networks, it might be the case that tuning the topology of the reservoir allows to obtain more efficient ESNs. Further work will address this research question, and two main directions can be imagined. The network can be built using different topological classes (e.g. small world, scale free, . . .); identifying classes of networks that are efficient for a given type of problem (i.e. such that randomly built networks from this class have a very high probability to solve the problem at hand) would indeed relieve the programmer from the task of optimizing the topology, restricting the search space to a fraction of the parameter space, where CMA-ES proved to be an efficient tool. It might be the case, however, that for reservoir computing, problem-specific topology tuning is nevertheless required anew for each problem. The main difficulty will then be to design efficient techniques for tuning the topology of large networks, as most existing methods do not really scale up to hundreds of neurons or more. Some hints have been recently given with Hyper-NEAT [16] on the one hand, and with the different approaches based on Genetic Regulatory Networks, starting with AGE, though other GRN approaches can be envisioned, too (see e.g. [2]).

5 Conclusion

Several recent studies have attempted to couple EAs with ESNs, mostly using supervised learning [15]. A limited number of works have used reinforcement learning to optimize ESNs with EAs, in which the network tasks were time series predictions [18,11] or robust spatial pattern formation ("flag" problems [3]). To our knowledge, our study is the first one to show the feasibility of the EA-ESN couple for motor control tasks. On addition, previous articles restricted evolutionary optimization to the reservoir weights [15] or more frequently the outgoing weights of the ESN. Here we show that optimizing additional ESN parameters could indeed be efficient.

In a supervised context, the results on a standard time series prediction problem reach the same precision when optimizing the output weights than the original results obtained using quadratic optimization, and further optimizations fail to improve this precision. For reinforcement learning tasks, the good news is that the Evolutionary Learning of ESNs works. Moreover, optimizing more than just the outgoing weights does improve the results. Furthermore, there seems to be a high dependency of the results on the topology of the reservoir, at least for the small sizes experimented with here. Hence, the results presented here do not satisfactorily answer the question of where ESNs stand between the two extremes of neuroevolution today: evolutionary optimization of the weights of a fully recurrent neural network (as proposed in [10]) and carefully crafted developmental systems that evolve the topology of highly efficient NNs for a given task [17,4]. Further experiments using more reliable test problems, and larger reservoir sizes, are needed to definitely address this issue. Additionally, a side take-home lesson from this paper concerns the usefulness of the double pole balancing problem as a benchmark for evolutionary control in general: the answer is clearly negative for us now (but had been claimed by others before), at last with the kind of fitness used up to now to tackle the problem.

Acknowledgments

The simulator of the double pole balancing problem relies on the source code kindly provided by Kenneth O. Stanley.

References

1. Auger, A., Hansen, N.: Performance evaluation of an advanced local search evolutionary algorithm. In: Proc. CEC 2005 (2005)
2. Banzhaf, W.: Artificial Regulatory Networks and Genetic Programming. In: Riolo, R., Worzel, B. (eds.) Genetic Programming Theory and Practice, ch. 4, pp. 43–62. Kluwer Academic Publishers, Dordrecht (2003)

3. Devert, A., Bredeche, N., Schoenauer, M.: Robust multi-cellular developmental design. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 982–989. ACM Press, New York (2007)
4. Dürr, P., Mattiussi, C., Floreano, D.: Neuroevolution with Analog Genetic Encoding. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 671–680. Springer, Heidelberg (2006)
5. Gomez, F.J., Miikkulainen, R.: Solving non-markovian control tasks with neuroevolution. In: IJCAI, pp. 1356–1361 (1999)
6. Gruau, F., Whitley, D., Pyeatt, L.: A comparison between cellular encoding and direct encoding for genetic neural networks. In: Koza, J.R., et al. (eds.) Proc. GP 1996, pp. 28–31. MIT Press, Cambridge (1996)
7. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004)
8. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolutionstrategies: the covariance matrix adaptation. In: Proc. CEC 1996, pp. 312–317. IEEE Press, Los Alamitos (1996)
9. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
10. Igel, C.: Neuroevolution for reinforcement learning using evolution strategies. In: Proc. CEC 2003, pp. 2588–2595. IEEE Press, Los Alamitos (2003)
11. Ishu, K., van der Zant, T., Becanovic, V., Ploger, P.: Identification of motion with echo state network. In: Proc. OCEANS 2004. MTTS/IEEE TECHNO-OCEAN 2004, vol. 3, pp. 1205–1210 (2004)
12. Jaeger, H.: The Echo State Approach to Analysing and Training Recurrent Neural Networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001)
13. Jaeger, H.: Tutorial on training recurrent neural networks. Technical report, GMD Report 159, Fraunhofer Institute AIS (2002)
14. Jaeger, H., Haas, H., Principe, J.C. (eds.): NIPS 2006 Workshop on Echo State Networks and Liquid State Machines (2006)
15. Schmidhuber, J., Wierstra, D., Gagliolo, M., Gomez, F.: Training recurrent networks by evolino. Neural Comput. 19(3), 757–779 (2007)
16. Stanley, K.: Compositional Pattern Producing Networks: A Novel Abstraction of Development. Genetic Programming and Evolvable Machines 8(2), 131–162 (2007)
17. Stanley, K.O., Miikkulainen, R.: Efficient reinforcement learning through evolving neural network topologies. In: Langdon, W.B., et al. (eds.) Proc. GECCO 2002, pp. 569–577. Morgan Kaufmann, San Francisco (2002)
18. Xu, D., Lan, J., Principe, J.: Direct adaptive control: an echo state network and genetic algorithm approach. In: Proc. IEEE International Joint Conference on Neural Networks IJCNN 2005, 31 July–4 August 2005, vol. 3, pp. 1483–1486 (2005)

Dynamic Cooperative Coevolutionary Sensor Deployment Via Localized Fitness Evaluation

Xingyan Jiang, Yuanzhu Peter Chen, and Tina Yu

Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
{xingyan,yzchen,tinayu}@cs.mun.ca

Abstract. We propose an innovative cooperative co-evolutionary computation framework, Dynamic Cooperative Coevolution (DCC), which provides dynamic coupling of neighboring species for the fitness evaluation of individuals. One feature of DCC is the utilization of local fitness to achieve a global optimum, which makes it possible for co-evolutionary algorithms to be applied in localized distributed environments, such as network computing. This work is motivated by our interest in autonomous sensor deployment, where a sensor can only communicate with those within a limited range. Our experiments show that DCC is effective in obtaining good solutions under such distributed and localized conditions.

1 Introduction

A wireless sensor network consists of a large number of sensor nodes distributed over an area of interest. Such networks are capable of observing and sensing the environment and sending the collected data to a data sink for further processing. Sensors must be deployed before they can transmit data. The deployment of static or mobile sensors, hence, is an important basis for sensor networking. A good placement yields high utilization of the network resources.

Two metrics are frequently used to evaluate the quality of sensor placement. The first one is *sensing coverage*, which is the area that the sensors in the network can monitor collectively. The second one is *energy consumption* during the sensor deployment. The energy cost in operating a sensor network includes moving nodes, sensing events in the environment, and transferring information. The lifetime of a sensor network is limited by the battery capacity of the nodes. In many applications where the replacement of battery is impossible, minimizing energy consumption during the sensors deployment is extremely important.

Autonomous sensor deployment has been studied using a variety of techniques. Howard et al. [2] described an incremental algorithm which deployed one sensor at a time. Each sensor node used the positions of previously deployed nodes to determine its own position. Zou and Chakrabarty [13] proposed a virtual force based algorithm to expand sensing coverage after the initial random deployment. The sensor movements were determined by the combined attractive and repulsive forces and the movements were coordinated by a cluster head. Wang et al. [9]

focused on repairing coverage holes when calculating sensors target positions using three Voronoi diagram based deployment protocols, VEC, VOR, and MiniMax. Chellappan et al. [1] proposed a flip-based algorithm to optimize both the coverage and the total number of flips. More recently, it has been demonstrated that computational intelligence techniques, such as fuzzy logic [8] and swarm intelligence [12] can be effective in sensor deployment.

In this paper, we propose *DCC*, a dynamic cooperative co-evolutionary framework, for autonomous sensor deployment. The algorithm facilitates sensors to construct partial network structures based on the local information exchanged by sensors within their neighborhood, i.e. *communication range*. Step by step, the global network structure is constructed to achieve the goal of *sensing coverage maximization* and *energy consumption minimization*. The paper is organized as follows. We first give a brief background of cooperative co-evolutionary algorithms in Section 2. In Section 3, the features of DCC are introduced, followed by a detailed description in Section 4. Simulation studies are presented with results analyzed in Section 5. Finally, we conclude this paper in Section 6.

2 Cooperative Co-Evolutionary Algorithms

Cooperative co-evolutionary algorithm (*CCEA*) is a special evolutionary algorithm proposed in [3,7]. Unlike the traditional EA [6], which solves a problem by searching the entire solution space, CCEA divides the problem into subproblems and searches the sub-solution spaces simultaneously. Since the sub-solution space is smaller, the algorithm may find better solutions faster.

In CCEA, multiple separate populations are created with their genotypic representations having no functional overlapping. Each population represents a different species and an individual therein represents a solution to the subproblem. Only the individuals of the same species can mate to produce offspring. However, the fitness of an individual is evaluated on the combination of its genotype and the representative genotypes of *all* other species. Each population evolves for a certain number of generations, which is equivalent to one *ecosystem generation*. At the end of each ecosystem generation, one representative is selected

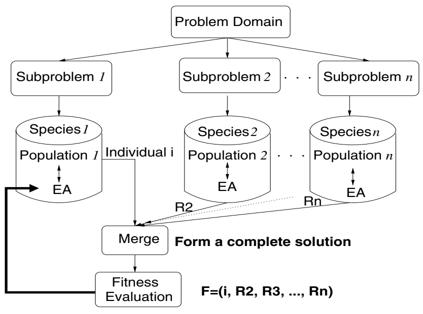


Fig. 1. A high-level view of CCEA

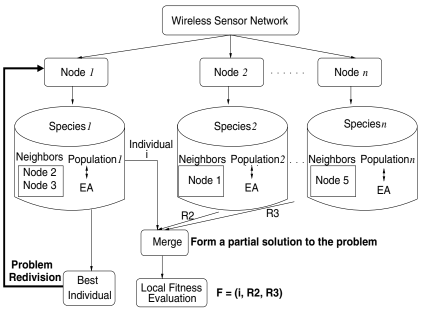


Fig. 2. High-level view of DCC

from each population and their genotypes are shared with other populations for fitness evaluation. The high-level flow of CCEA is given in Fig. 1, where R_i is the representative of species i.

There are researchers investigating *problem decomposition* and the efficiency of single-best collaboration during the evolution. Wiegand and colleagues [11] argued that when a problem is divided in such a way that there exists contradictory cross-population epistasis (inter-dependency), single-best collaboration would not produce good solution. To address the inter-dependency issue, Weicker and Weicker [10] proposed dynamically merging the species when inter-dependency of variables in cross populations was detected. Kim and Ryu [5] went farther by allowing not only merging but also splitting the species when the inter-dependency no longer exist during the evolution. Our cooperative co-evolutionary framework also provides dynamic division of species. The main features of the framework are described in the following section.

3 Dynamic Cooperative Coevolution Framework

DCC is a completely localized distributed algorithm in that each population only collaborates with populations within its neighborhood for fitness evaluation. This is an essential requirement for distributed computing where every node in the system only has a local view of the environment. Global broadcasting of messages is possible but is considered infeasible due to the high computation overhead required. To work with such constraints, the following mechanisms have been developed so that co-evolutionary algorithms can be applied effectively in localized and distributed environments, such as network computing. The DCC framework is depicted in Fig. 2.

1. **Flexible and dynamic problem division.** Under distributed environments where the location of each node may change dynamically, the partitioning of the problem (i.e. the sub-solution that each population evolves) also changes. This is contrast to the CCEA where the solution each population evolves is fixed throughout the execution of the algorithm. One consequence of this dynamic problem division is that the populations that collaborate for fitness evaluation also change during the algorithm execution.
2. **Energy efficient partial fitness evaluation.** Because each population can only assume the availability of local information within its proximity, the fitness evaluation must tolerate the missing input from beyond the neighborhood. This is a salient contrast to CCEA, where fitness cannot be evaluated without the information from all other populations.
3. **Two operation modes for effective and efficient evolutionary search.** In spirit, the first mode (mode I) is similar to the *splitting species* proposed in [5] and the second mode (mode D) is similar to the *merging species* proposed in [10]. If evolutionary search reaches a local optimum, merging species helps escaping the local optimum and making the search more effective. If evolutionary search reaches the basin of a global optimum after escaping a local optimum, splitting species helps the search find the global optimum faster. We

developed a simple method to detect that a population might have reached a local optimum by checking the existence of coverage holes in the neighborhood. If one or more holes exist, operation is switched to mode D for 1 ecosystem generation cycle. Alternating these two modes can accelerate the search process while avoiding local optima.

4 Algorithm Design for Autonomous Sensor Deployment

We have implemented the DCC concept to solve the autonomous sensor deployment problem[1]. DCC consists of 3 major stages: *planning*, *computing*, and *moving*. A complete pass of the 3 steps is called an *ecosystem cycle*. In the planning stage, a sensor first divides the problem and prescribes a search space within its proximity in which it will find a target position and move to it at the end of the current ecosystem cycle. In the computing stage, the sensor executes a local EA within its search space to calculate the best target position using a fitness calculated from local information. Finally it moves to the target position in the moving stage. Once the movement is completed, the new search space for each sensor is calculated. The sensor may switch its operation mode (described in the following paragraph) if needed and then starts a new ecosystem cycle to search for the next position that the sensor would move to next. This process repeats many times until the specified number of ecosystem cycles is reached. Fig. 3 gives the high-level flow of the implementation.

After the sensors are randomly distributed in the field initially, one *local population* is used to evolve one sensor's target position. Each local population can be executed using one of two operation modes: mode I (Independent) evolves only a sensor's position and mode D (Dynamic) evolves the positions of a sensor and

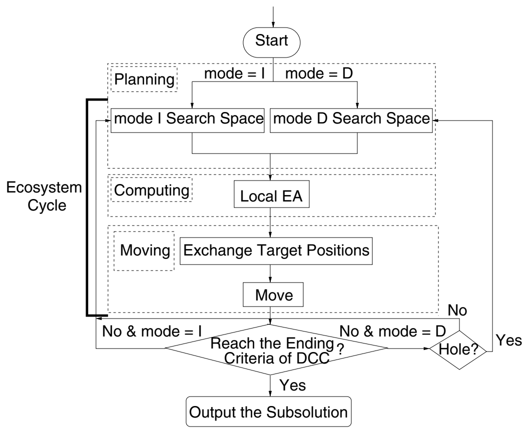


Fig. 3. DCC flow chart

[1] The source code is located in http://www.cs.mun.ca/∼xingyan/ppsn/DCC.tar.gz

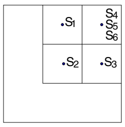


Fig. 4. An example of local optimum

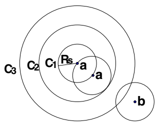


Fig. 5. Potential movement and overlaps

its neighboring sensors. In the first case, the fitness of an individual is evaluated on the combination of its genotype and the representative genotypes from the neighboring sensor populations. In the latter case, the fitness of an individual is evaluated on its own genotype, which contains the positions of a sensor and its neighboring sensors. Regardless of the operation mode, the fitness of an individual only covers a partial network of the entire sensor network.

Under mode I, the search space of a local population is two-dimensional: the *x, y* location of a sensor. With each local population searching a 2-dimensional space separately and simultaneously, the global sensor network can be obtained reasonably fast. However, occasionally sensors may get stuck in a local optimum. For example, in Fig. 4, $S_1, S_2, \ldots, S_6$ are 6 sensors used to cover an area, where S_4, S_5 and S_6 have identical location[2]. It is obvious that the sensing coverage would increase if some sensors move to the left or the lower region of the field. However, this would never happen because the current sensor locations give the best coverage (the union of the sensing region of all sensors), based on the neighboring sensor positions provided at the beginning of the ecosystem cycle. In order to obtain locations that give a better coverage than the current ones do, the neighboring sensors need to have different locations. Mode D provides this flexibility by allowing both the locations of a sensor and its neighboring sensors to evolve and helps the populations escape the local optimum.

In mode D, the search space of a local population is multiple-dimensional: the *x, y* locations of a sensor and its neighboring sensors. Unlike mode I where the neighboring sensor locations that are used for fitness evaluation are fixed throughout the ecosystem cycle, the neighboring sensor locations also evolve. It models the potential local interactions and uses that to improve the local estimate of fitness. Note that the evolved neighboring sensor positions are only used for fitness evaluation. They have no impact on the neighboring sensors' new positions, which are only decided by the "fittest" individual in the neighboring sensor populations.

The implementation is based on the following assumptions: 1) each sensor knows its own location. 2) a sufficient number of sensors are deployed so that they can potentially cover the entire area. 3) each sensor has a sensing range, R_s, a communication range, R_c, and $R_c \geq 3R_s$. DCC algorithm executes a sequence of ecosystem cycles, where each cycle consists of 3 steps: planning, computing, and moving. We explain each step in the following sub-sections.

[2] We use a square area to indicate a sensor's sensing region for simplicity.

4.1 Planning: Problem Division

At the beginning of each ecosystem cycle, the entire deployment area is partitioned based on the current sensor locations in the network: each sub-area is the *sensing region* of a sensor, i.e. the circle of radius R_s centered at the position of the sensor. A local evolutionary algorithm is executed for each sensor to locate a new position within the region where the sensor will move to at the end of the cycle. Under the assumption that $R_c \geq 3R_s$, the new coverage of a sensor and its non-neighboring sensors would never overlap no matter where they move to. For example, in Fig. 5 node a has a communication range $R_c = 3R_s$ and centered at itself are three circles of radii R_s, $2R_s$ and $3R_s$, denoted by C_1, C_2, and C_3, respectively. The search space restricts node a to move within C_1, which implies that its new coverage will be restricted to C_2. For a non-neighboring node b, which is out of C_3, its sensing coverage will not overlap with the new coverage of node a, no matter where it moves to within the range of its search space. This restriction is important for the fitness evaluation described in Section 4.2.

For each local population, the individual with the highest fitness at the end of each cycle is selected and the sensor position information is exchanged with all its neighboring sensors (i.e., those within its communication range) populations through a reliable wireless communication channel. At the initial cycle where individuals in the population were randomly generated, representatives are selected randomly.

4.2 Computing: New Position Exploration

This section describes each component of the evolutionary algorithm.

Representation. We used a fixed length array of n elements to represent the genotype of an individual, where n is the total number of sensors in the network. Each element i ($i = 1, 2, \ldots, n$) is the position $\{x_i, y_i\}$ of sensor i in the deployment area (see top diagram of Fig. 6). Since a sensor only has position information of its neighboring sensors, the elements in the genotype corresponding to non-neighboring sensors contain invalid values. To distinguish a neighboring sensor from a non-neighboring one, a second non-evolvable chromosome of length n is used (see bottom diagram of Fig. 6). There, a value 0 indicates that the corresponding element in the first chromosome is a non-neighbor while 1 indicates that it is a neighbor and $\star$ indicates the sensor itself. This 2-chromosome genotype representation provides the flexibility to facilitate the dynamic problem division explained in Section 3. When a sensor is switched from being a neighbor to a non-neighbor (or vice visa) for a particular sensor after movement, an update of the second chromosome can reflect such change.

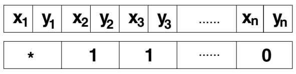


Fig. 6. The 2-chromosome genotype representation

Fitness Evaluation. The fitness of an individual (sensor position) is determined by the total sensing coverage induced by the position and the travel distance between this and the current position of the sensor. Assume the sensing region of node i is A_i ($i = 1, 2, \ldots, n$), each A_i is a subset of the entire deployment area U, i.e. the universe. For a given node, the sensing coverage is the union of its sensing area and the sensing areas of its neighboring sensors. Let $\mathcal{H} = \langle h_1, h_2, \ldots, h_n \rangle$ be the second chromosome of the sensor's genotype. To calculate its coverage, we define a companion vector $\overline{\mathcal{H}} = \langle \overline{h_1}, \overline{h_2}, \ldots, \overline{h_n} \rangle$, where $\overline{h_i} \in \{\emptyset, U\}$, for each $\mathcal{H}$. Specifically, $\overline{h_i} = U$ if $h_i \in \{1, \star\}$ and $\overline{h_i} = \emptyset$ if $h_i = 0$. Thus, the coverage unioned over the neighborhood of a sensor is:

$$\bigcup_{i=1}^{n} \left(\overline{h_i} \cap A_i \right)$$

For an individual with sensor position which is d away from the current position, its fitness F is:

$$F = \left| \bigcup_{i=1}^{n} \left(\overline{h_i} \cap A_i \right) \right| - w \times d,$$

where w is a weight parameter to adjust the tradeoff between coverage and movement. Although the fitness evaluation of DCC only uses local information from its neighboring nodes, it will be shown (see Section 5) that the computed fitness value is able to drive the evolutionary search to find target positions that give good overall coverage and requires a small amount of energy consumption.

Selection and Genetic Operations. Among a population of P individuals, the $|Q|$ fittest are selected as parents, denoted by Q, to reproduce the same number of offspring Q' via arithmetic crossover. The Q individuals are paired based on their ranks: the first rank is paired with the second rank, the second rank is paired with the third rank and so on. The arithmetic crossover takes the average of the two parents' gene values as the gene value of its offspring.

Out of $P \cup Q'$, the $|P|$ fittest individuals survive and are carried over to the next generation. This process continues for g generations and the fittest individual at the end is the target position where the sensor moves itself to.

4.3 Moving: Automatic Sensor Relocation

Once the new position of a sensor is determined, the sensor moves to that location automatically using its actuation component. Then it broadcasts its new position and prepares for the next cycle. In some network scenarios, the assumption of $R_c \geq 3R_s$ can not be satisfied. In this case, the local coverage can not be calculated precisely. To alleviate this situation, an additional broadcast of the new location is necessary before the sensor starts to move to the new location. Further, a limited-scope flooding could be used alternatively.

5 Experimental Analysis

We used the implemented DCC algorithm to simulate the autonomous sensor deployment under various initial conditions: sensors are distributed uniformly

Table 1. Simulation parameters

Parameter	Setting	Parameter	Setting
Deployment area size U	$100^2, 200^2, 300^2(\text{m}^2)$	Sensing range R_s	20m
No. of sensor nodes n		Communication range R_c	60m
area 100^2m^2	10, 12, 14, 16;	Population size $\|P\|$	10
area 200^2m^2	40, 50, 60, 70;	No. of offspring $\|Q\|$	5
area 300^2m^2	70, 80, 90, 100	No. of runs	30
No. of eco cycles g_e	30	No. of gen in each eco cycle g	5

to three different sizes of field: $100 \times 100\text{m}^2$ (small), $200 \times 200\text{m}^2$ (medium) and $300 \times 300\text{m}^2$ (large). For the small size field, 10, 12, 14 and 16 sensors are deployed; for the medium size field, 40, 50, 60 and 70 sensors are deployed; for the large size field, 70, 80, 90 and 100 sensors are deployed. Table 1 summarizes the parameter values used to carry out our simulation.

We use 3 metrics to evaluate the experimental results averaged over 30 runs: *moving distance*, *convergence time* and *sensing coverage*. Moving distance is the average distance that a sensor in the network has to travel from its initial to final position. Convergence time is the number of ecosystem cycles it takes for *all* sensor populations to converge, i.e. the best individual fitness stopped improving. Sensing coverage is the percentage of the deployment field that is covered by the deployed sensors. Also, to select a weight parameter (w) that balances the evolutionary force toward solutions that give *large* coverage and *small* moving distance, we conducted a preliminary study and chosen $w = 1$ [4].

5.1 Simulation under Mode I Only

We study mode I performance under different network sizes (small, medium, large) using a different number of sensors as that given in Table 1. Fig. 7 shows that the global network coverage improves rapidly during the first few ecosystem cycles and the populations converge around generation 7. Fig. 8 gives the global network coverage and the moving distance over time for one run on a medium size field. It shows that the coverage increases while the moving distance decreases as the evolution progresses. The selected w (1) is able to balance the two conflicting objectives and direct the evolutionary search to find a good solution.

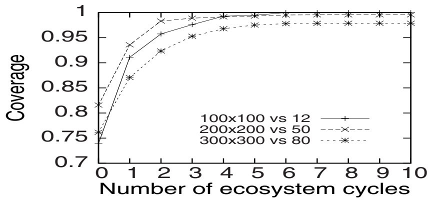


Fig. 7. Coverage improvement

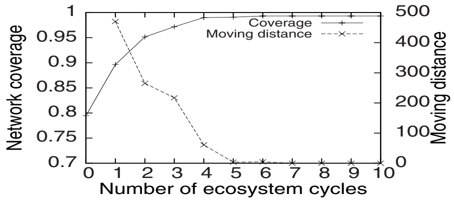


Fig. 8. Coverage vs. moving distance

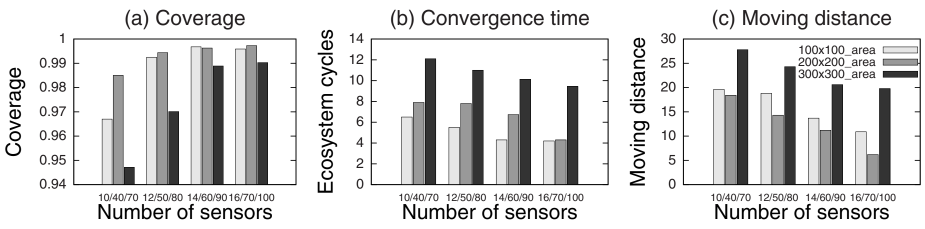


Fig. 9. Performance of Mode I evaluated by 3 different metrics

When the best individual in all populations stopped improving, the 3 metrics were evaluated (see Fig. 9). The general observation from these experiments is that, as the sensor nodal density increases, so does the induced network coverage, while the convergence time and moving distance decrease. This is reasonable as a larger number of sensors in the network makes it easier to cover a wider area of the deployed field under a smaller amount of time and moving distance.

5.2 Simulation under the Alternation of Mode I & D

To investigate the benefit of mode D in helping the populations escape local optima and deliver better solutions, we carried out two sets of experiments: one operated mode I only and the other alternated mode I & D with 5 and 1 ecosystem cycles intervals, i.e. 5 mode I cycles followed by 1 possible mode D cycle. This alternation was selected because a population is not likely to reach a local optimum during the first 5 cycles, hence should be operated under mode I. At the end of the 5th cycle, the best individual in each population is checked for *coverage holes* (an area that is not covered by any sensor in its neighborhood). If there is any hole, the local GA is switched to mode D for 1 cycle and switched back to mode I the following cycle, since mode I runs faster than mode D (see Section 4). This check is carried out for each sensor population. The average coverage of 30 runs and the numbers of runs achieving 100% coverage are given in Table 2. Overall, both setups provide very good coverage. Nevertheless, alternating mode I & D delivers a higher number of runs that produced 100% coverage.

To validate our hypothesis that mode D improves performance by helping the populations escape local optima, we conducted another experiment with

Table 2. Coverage Comparison Between Mode I and Mode I & D

	Mode I		Mode I & D	
sensors	coverage	100% cover	coverage	100% cover
40	98.50%	0	99.33%	1
50	99.44%	0	99.88%	15
60	99.63%	0	99.98%	25
70	99.73%	0	99.99%	27

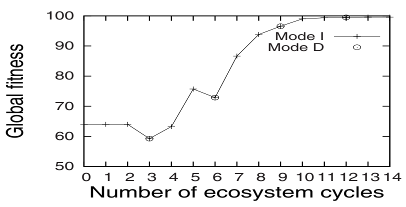


Fig. 10. Global fitness under Mode D & I

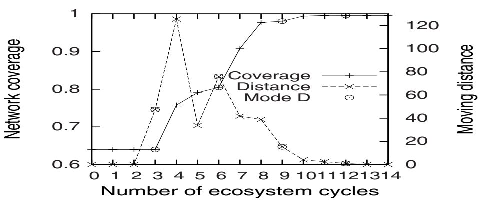


Fig. 11. Coverage vs. moving distance

10 sensors initialized to locations that give a local optimum coverage (64%) and deploying them to a medium size field. The simulation was carried out by alternating 2 cycles of mode I followed by 1 possible cycle of mode D. The best global fitness (see Fig. 10) shows that after 2 cycles of no fitness improvement, the fitness declined after the execution of mode D, which is caused by a large moving distance (see Fig. 11), indicating the sensor has escaped the local optimum. After that, the global fitness starts to climb and eventually reaches 100% coverage.

6 Conclusions

We have presented an innovative cooperative coevolutionary framework, DCC, for optimization tasks in localized and distributed environments. By supporting *dynamic problem division*, *partial fitness evaluation* and *2 operation modes*, DCC is shown to be effective in the autonomous sensor deployment task, where high coverage and low energy consumption were achieved in a short period of time.

References

1. Chellappan, S., Bai, X., Ma, B., Xuan, D.: Sensor networks deployment using flip-based sensors. In: Proceedings of IEEE MASS (November 2005)
2. Howard, A., Mataric, M.J., Sukhatme, G.S.: An incremental self-deployment algorithms for mobile sensor networks. Autonomous Robots, Special Issue on Intelligent Embedded Systems 13(2), 113–126 (2002)
3. Husbands, P., Mill, F.: Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling. In: Proceedings of ICGA, pp. 264–270 (1991)
4. Jiang, X., Chen, P.Y., Yu, T.: Localized Distributed Sensor Deployment via Co-evolutionary Computation. In: Proceedings of the IEEE International Conference on Communications and Networking (2008)
5. Kim, M.W., Ryu, J.W.: An efficient coevolutionary algorithm using dynamic species control. In: Proceedings of ICNC (2007)
6. Mitchell, M.: An introduction to genetic algorithms. MIT Press, Cambridge (1996)
7. Potter, M.A.: The design and analysis of a computational model of cooperative coevolution. PhD thesis, George Mason University (1997)
8. Shu, H., Liang, Q., Gao, J.: Distributed sensor network deployment using fuzzy logic systems. International Journal of Wireless Information Networks 14(3), 163–173 (2007)

9. Wang, G., Cao, G., Porta, T.L.: Movement-assisted sensor deployment. In: IEEE INFOCOM (March 2004)
10. Weicker, K., Weicker, N.: On the improvement of coevolutionary optimizers by learning variable interdependencies. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1627–1632 (1999)
11. Wiegand, R.P., Liles, W.C., De Jong, K.A.: The effects of representational bias on collaboration methods in cooperative coevolution. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 257–268. Springer, Heidelberg (2002)
12. Wu, X., Cho, J., d'Auriol, B.J., Lee, S.: Mobility-assisted relocation for self-deployment in wireless sensor networks. IEICE Trans. 90-B(8), 2056–2069 (2007)
13. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization based on virtual forces. In: Proceedings of IEEE INFOCOM, pp. 1293–1303 (2003)

On the Run-Time Dynamics of a Peer-to-Peer Evolutionary Algorithm

J.L.J. Laredo[1], A.E. Eiben[2], M. van Steen[2], and J.J. Merelo[1]

[1] Department of Architecture and Computer Technology
University of Granada, Spain
{juanlu,jmerelo}@geneura.ugr.es
[2] Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
{gusz,steen}@cs.vu.nl

Abstract. In this paper we propose an improvement on a fully distributed Peer-to-Peer (P2P) Evolutionary Algorithm (EA) based on autonomous selection. Autonomous selection means that individuals decide on their own state of reproduction and survival without any central control, using instead estimations about the global population state for decision making. The population size varies at run-time as a consequence of such a decentralized reproduction and death of individuals. In order to keep it stable, we propose a self-adjusting mechanism which has been shown successful in three different search landscapes. Key are the estimations about fitness and size of the population as provided by a gossiping algorithm. Such an algorithm requires several rounds to collect the information while the individuals have to wait for synchronization. As an improvement, we propose a completely asynchronous EA which does not need waiting times. The results show that our approach outperforms quantitatively the execution time of the synchronous version.

1 Introduction

Spare cycles among interconnected nodes constitute a free and powerful source for high performance computing, Peer-to-Peer (P2P) systems form an alternative to jointly constitute a single virtual computer. Nowadays, there are successful cases of virtual supercomputers based on volunteers sharing their CPU idle cycles (e.g. the BOINC project [1]).

However, Evolutionary Computing has just recently entered this arena and there are still many challenging issues. The DREAM project was one of the pioneers on distributed P2P EAs coming up in [2] with the equally named DREAM framework. Despite the P2P approach, the island-based parallelization of DREAM was shown in [9] to be insufficient for tackling large-scale decentralized scenarios.

Two of our most recent works, [12] and [8], have moved the focus from distributed P2P EAs into finer-grained approaches within the field of spatially structured EAs. As stated in [11], a spatially structured EA can be modeled as a graph in which the vertices are individuals and edges represent relationships between them. Obviously, a graph can be easily mapped to a network topology and consequently a spatially structured EA can be easily distributed.

In this paper, we analyse a distributed P2P EA in which the population structure is defined by a P2P overlay network. The network keeps small-world properties by means of the gossiping protocol Newscast [6]. Such a kind of small-world graphs have been shown, by Giacobini et al. in [4], to be suitable as population structure for an EA, outperforming *panmictic* approaches.

But population structure is not the only issue in a P2P EA. The absence of a central control requires a mechanism to convey global estimations into each local individual. This way, individuals can make local decisions about their status in a decentralized evolution. In order to get estimates about population fitness and size, we follow the counting algorithm proposed in [12] which deploys the aggregation protocol described in [5]. It consists of an iterative gathering of information that after several time steps (or rounds) becomes accurate enough to proceed with local decision making.

Making a local decision in EAs is not straightforward when the parallelization grain is a single individual. Despite crossover and mutation operators requiring one or two individuals, selection involves all of them, or at least a few (such as in tournament selection).

In fact, the autonomous selection presented by Eiben et al. in [3] uses locally available information about global estimations (e.g. those provided by the counting algorithm) and determines the selection probabilities for each individual with a locally executable function based on its own fitness against averaged global fitness. Subsequently, the fittest individuals survive for reproduction while the worst are erased from population. The consequence of such a decentralized process may lead to a run-time population resizing which could get out of control.

In [12], the authors overcame the issue of population size implosions/explosions using an self-adjusting mechanism for controlling the parameters of a sigmoid function (i.e. the seminal function for autonomous selection). Unfortunately, preliminary experiments in different search landscapes have shown sigmoid to be very sensitive under different roughness conditions. Hence, it turns out that keeping the population size under control requires of a hand-made calibration of the self-adjusting mechanism.

Our proposal focuses on the following improvements over the work presented in [12]:

1. We propose a self-adjusting mechanism able to keep the population size stable in different search landscapes. Instead of the sigmoid, we use a simple linear function and a single adjustable parameter, ρ, which self-regulates the local selection pressure by controlling the function slope.
2. In order to get accurate global estimations, the counting algorithm spends several rounds in which the adaptation stage has to wait. In spite of such a necessary synchronization for autonomous selection, we propose an asynchronous EA that uses the counting rounds to evolve a population of individuals' replicas. Each replica evolves with those in its neighbourhood using

tournament selection. If the original individual survives the autonomous selection process, the evolved replica replaces it. As a consequence of reducing the ratio of rounds per evaluations, the execution time of the algorithm is improved.

The efficiency of our approach quantitatively outperforms previous large-scale distributed EAs. The key is the combination of local evolution of individuals' replicas and global resynchronization of the decentralized EA.

The rest of the paper is structured as follows. The overall model is presented in Section 2. We propose, in Section 3, a test suite composed of three real-coding functions with different roughness degree. In Section 4, we deduce from the run-time dynamics that the accuracy of the estimations is good enough to keep the population size stable. Finally, we reach some conclusions and propose some future work lines in Section 5.

2 Proposed Model

Algorithms 1 and 2 show respectively the pseudo-code of the algorithm and the work-flow of an iteration.

Algorithm 1. Outline of the self-adjusting distributed evolutionary algorithm

```
initialize $individual$
$individual_{replica} \Leftarrow individual$
repeat
    if adaptation stage then
        exchange information by gossiping
        estimate population size, average fitness and best fitness
        evolve $individual_{replica}$ within the neighbourhood
        update selection parameters by adaptation
    end if
    if resetting stage then
        if $individual$ is not able to survive then
            die
        else
            $individual \Leftarrow individual_{replica}$
            new $individual \Leftarrow reproduction(individual + random \quad individual)$
        end if
    end if
until die or another stop criterion
```

During the adaptation stage ($n+1$ first rounds), each $individual_{replica}$ evolves within its neighbourhood using tournament selection. This stage is required for estimations over the decentralized population. In the resetting stage ($n+2$ time step), the fittest *individuals* survive, acquire the evolved genome of their own $individual_{replica}$ and generate a new *individual*. Additionally, the values of the counting algorithm are reset and the number of rounds (n) is estimated for the next iteration.

2.1 Counting Algorithm

The counting algorithm described in algorithm 3 was presented in [12] and extends a P2P aggregation mechanism described in [5]. It provides estimations

Algorithm 2. Outline of an iteration of the distributed algorithm

```
1 to n time steps: Gossiping rounds
  Exchange information with neighbours
  evolve $individual_{replica}$ within the neighbourhood
  Perform the counting algorithm
n + 1 time step: Adaptation
  Call the adaptation process
  Update the parameters for selection
n + 2 time step: Resetting
  Call survive process
  Either die or $individual \Leftarrow individual_{replica}$
  Reset the values for the counting algorithm
  Calculate steps needed (n) for next iteration
```

about the current best fitness, average fitness and total size of the population to each individual. The information is iteratively flooded among the nodes (individuals) and after several iterations (rounds) estimations are available to the nodes. The number of rounds needed is estimated as a logarithmic function of the total size of the population, growing as the size increases (i.e. a population size of 100 individuals would need 12 rounds for estimations while 1000 would need of 18).

Algorithm 3. Counting algorithm

```
initially
msg_tag ← id /*all nodes have unique identifier*/
size_est ← 1 /*initially a node knows that only it exists*/
avg_est ← fitness_value
compute $estimates(size\_est, avg\_est, msg\_tag)$
repeat
    pull $estimates(size\_est_p, avg\_est_p, msg\_tag_p)$ from neighbor p
    if (msg_tag < $msg\_tag_p$ ) then
        /*abort own counting process*/
        msg_tag ← $msg\_tag_p$
        size_est ← 0
    else if (msg_tag > $msg\_tag_p$ ) then
        /*abort other counting process*/
        $size\_est_p$ ← 0
    end if
    $size\_est_p \leftarrow \frac{size\_est + size\_est_p}{2}$
    $avg\_est_p \leftarrow \frac{avg\_est + avg\_est_p}{2}$
    push $estimates(size\_est, avg\_est, msg\_tag)$ to neighbor p
until desired number of gossiping rounds
```

Each node starts an estimation process for the system values but just the process with the highest identifier survives.

2.2 Adaptation and Survival

The autonomous selection determines the probability of survival for each individual by the following equation:

$$P(x) = linear_{\rho}(\Delta f(x)) = \frac{1-\rho}{\Delta f(x_{best})}\Delta f(x) + \rho \quad (1)$$

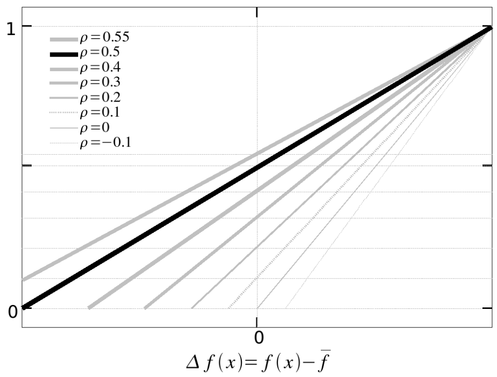


Fig. 1. Linear function for different values of the adjustable parameter ρ

where $\Delta f(x)$ is the deviation of the fitness with respect to the average fitness, $\Delta f(x) = f(x) - \overline{f}$. The function assigns a probability of survival equal to 1.0 for the best individual $P(\Delta f(x_{best})) = 1.0$.

Additionally, the slope of the linear function is determined by the average fitness $\overline{f}$, with $\Delta\overline{f} = \overline{f} - \overline{f} = 0$, where the probability of survival is ρ, $(P(\Delta\overline{f}) = \rho)$.

ρ is an adjustable parameter within the range $[-0.1, 0.55]$. This range has been empirically calibrated in preliminary experiments to prevent population implosions/explosions. The self-adjusting procedure is shown in algorithm 4.

Algorithm 4. Outline of the self-adjusting procedure

```
$P \Leftarrow$ Initial Population Size
$\rho \Leftarrow 0.5,\ \rho \in [-0.1, 0.55]$
repeat
  if adaptation stage then
    $P_{estimated} \Leftarrow$ Counting Algorithm
    if $P_{estimated} > P$ then
      $\rho = \rho - 0.1$
    else if $P_{estimated} < P$ then
      $\rho = \rho + 0.1$
    end if
  end if
until stop criterion
```

Initially $\rho = 0.5$, which would probabilistically maintain the population size if we assume normality conditions in the fitness distribution. If the population size is bigger than the initial population, ρ is decreased by 0.1, otherwise ρ is increased by 0.1 (such a value has been empirically calibrated). From the different values of ρ (as shown in Figure 1) the algorithm self-adjusts the ratio of survival by changing the selection pressure.

3 Experimental Setup

In order to test the run-time dynamics of the algorithm, we have conducted experiments in the P2P simulator PeerSim [7]. We have chosen as a benchmark three real-coding test functions from the test suite proposed by Suganthan et al. in [10] (Shifted Sphere function, Schwefel function and Shifted Rotated Rastrigin's function). This set includes different search landscapes derived from a sphere, the Schwefel problem and the Rastrigin multimodal function. It is important to note that our research objective is not to outperform existing results. Instead, we are initially interested in exploring the extent in which fully decentralized solutions can be successful. Therefore, to consider an EA run successful we allow an error margin of 1 for all test functions. Additionally, we have set the size of the problem instances to a medium degree of difficulty for a GA (i.e. chromosome sizes are 30, 10 and 10 respectively).

As a baseline for comparison, we have used the distributed P2P EA proposed in [12] to which we will refer as synchronous version from here on, an asynchronous version using a fixed population size and tournament selection and a standard generational 1-elitism GA. The adaptation stage of the synchronous version has been set with the self-adjusting mechanism proposed in Section 2.2, the rest of the parameter setup is shown in Table 1.

Table 1. Parameters of the algorithms

Initial Population Size	200 individuals
Recombination	BLX-0.5 Crossover, $p_c = 1.0$
Mutation	BGA, $p_m = 0.01$
Initial value of ρ	0.5
Termination condition	optimum found with the required accuracy or 100000 evaluation spent or population size = 0 or population size > 600
Selection Parents (original)	Autonomous Selection
Selection Parents (replica)	Binary Tournament + individual

4 Experimental Results

Figure 2 shows the dynamics of the counting algorithm and the self-adjusting mechanism in the control of the population size.

On one hand, the counting algorithm provides accurate estimations about the population size every n rounds (estimations are shown as circles). On the other hand, the self-adjusting mechanism keeps the population size stable by fluctuating around the pre-established initial size. From the observation we can see how the peaks grow when the algorithm is getting close to the problems' optima (the convergence is represented in Figure 3). The most probable hypothesis for these peaks is that the distribution of Δfs is biased by the global optimum. Hence, the distribution would lose normality conditions as it is getting close to the optimum with the consequent lack of effectiveness in the self-adjusting mechanism.

There is an important difference between our proposal and the synchronous version. In our approach, the population size is fixed during the evolution of individuals' replicas. Afterwards, the population size adjusts in the resynchronization period. Figure 2 shows that the population size does not explode/implode, which is coherent with the previous formulated hypothesis: Once that the algorithm is approaching the problem optimum, local evolution yields success before

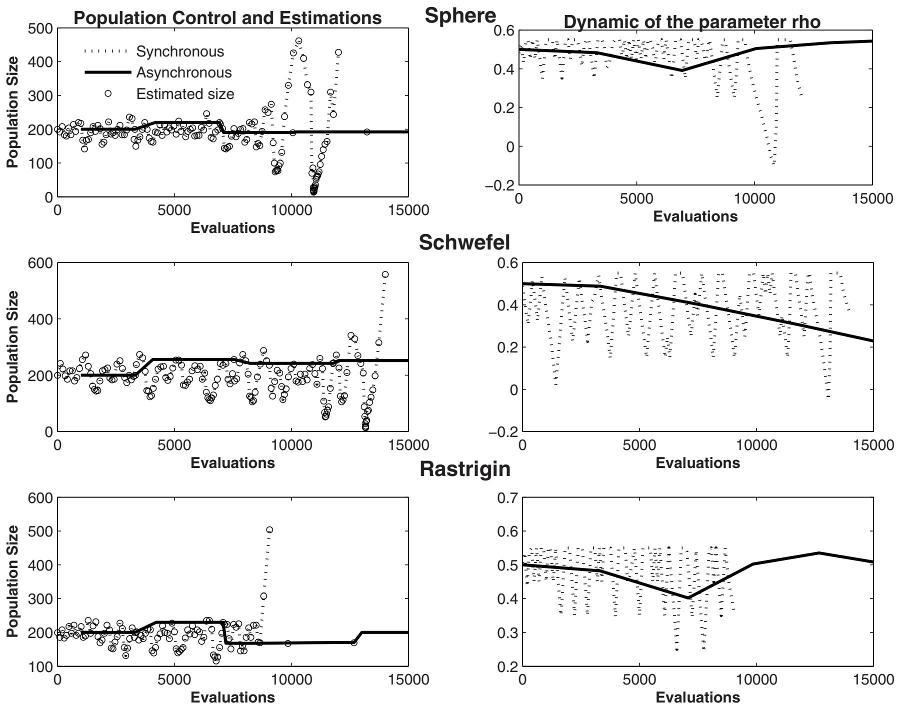


Fig. 2. Dynamic of the population size in the synchronous and asynchronous versions during one run *(left)* and respective values from ρ *(right)*. From *top* to *bottom* the run-time adjustment for the three functions. Circles represent the averaged estimation of the population size provided by the counting algorithm.

an explosion in the population size. In fact, such an hypothesis will have to be validated in future works.

Not only is our method able to keep in check the population size, but it is also able to find the optimum with the require accuracy. Figure 3 depicts the convergence curves of the best and average fitness of our proposal and the respective estimations by the counting algorithm. The counting algorithm shows a very accurate estimation for the best fitness while the estimated average fitness is not so accurate. The reason is that the decentralized aggregation of the average fitness is a more complex process than a simpler flooding of the best solution by gossiping. Nevertheless, it is important to note here that despite the estimation errors, the algorithm is robust enough to converge to the problem solutions.

Finally, Figure 4 shows the best fitness curves of our asynchronous approach, the globally synchronized one presented in [12], the distributed version using a fixed population size and tournament selection and the standard GA. Each curve depicts the number of rounds needed to improve the fitness (i.e. number of cycles in simulator driven experiments). All approaches reach success criteria, however,

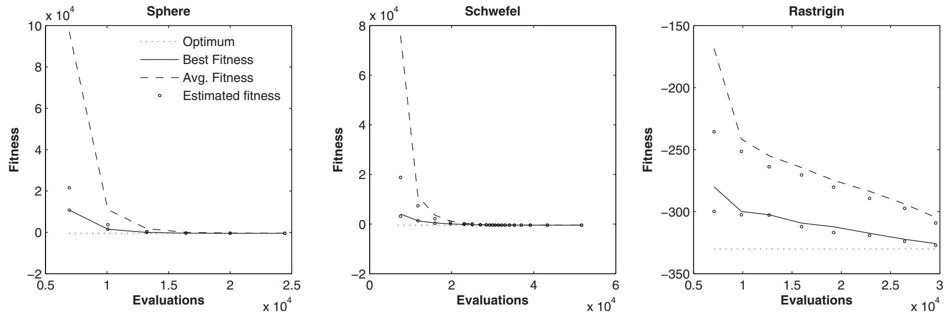


Fig. 3. Convergence curves in the sphere, Schwefel and Rastrigin test functions. Circles represent the averaged estimation of the best and average fitness.

our proposal needs $\sim$ 90% less time running than the synchronous approach and $\sim$ 99% less than the standard GA which follows a sequential approach. In fact, most of the cycles in [12] are employed in the counting algorithm, being useless from an evolutionary point of view. Therefore, the improvement consists in making those idle cycles useful, we use the local evolution of individuals' replicas to that end. With respect to the distributed approach using a fixed population size, the times are equivalent. Nevertheless, the main drawback of the fixed population size approach is not being resilient to nodes failures.

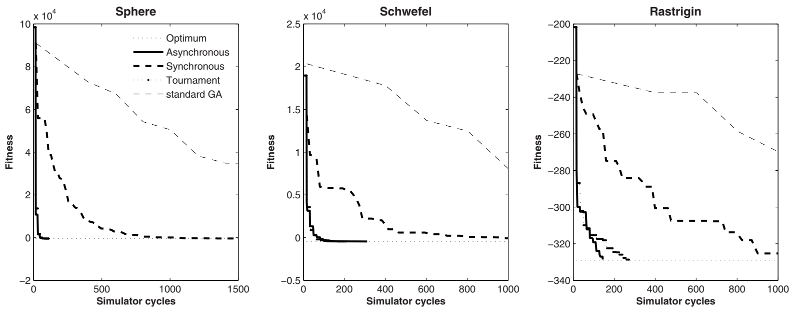


Fig. 4. Best fitness convergence curves for the asynchronous and synchronous versions, a distributed approach using a fixed population size and a standard GA. From left to right, the sphere, Schwefel and Rastrigin test functions. On the *x-axis*, simulator cycles stand for the number of rounds that best fitness needs to improve.

5 Conclusions and Future Works

In this paper we have proposed an asynchronous and distributed Peer-to-Peer Evolutionary Algorithm. The whole process is tackled in a decentralized manner in which every individual decides on its own state of reproduction and survival based on autonomous selection and global estimations. The variability on the population size is adjusted by a self-adjusting mechanism that maintains the size around the initial given value. In order to study the run-time dynamics of the algorithm, we have proposed a test suite of three different search landscape with different roughness degree. For all test functions our new EA (using adaptively controlled selection) was able to find the optimum with the required accuracy.

The proposal also includes an asynchronous replica mechanism which avoids the global synchronization presented in [12]. The execution time has been clearly outperformed which is key in a parallel environment. Therefore, we conclude that our proposal is a feasible approach towards a fully decentralized EA with a special focus on P2P.

There are still many challenge to tackle concerning P2P EAs that will have to be studied in future works. We plan to dive in the algorithmic performance and compare our method with other spatially structured algorithms (not focused in P2P necessarily). Additionally, a study on a real environment would provide feedback on actual problems of full decentralization.

Acknowledgements

This work has been supported by the Spanish MICYT project TIN2007-68083-C02-01, the Junta de Andalucia CICE project P06-TIC-02025 and the Granada University PIUGR 9/11/06 project.

References

1. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@home: an experiment in public-resource computing. Commun. ACM 45(11), 56–61 (2002)
2. Arenas, M.G., Collet, P., Eiben, A.E., Jelasity, M., Merelo, J.J., Paechter, B., Preuss, M., Schoenauer, M.: A framework for distributed evolutionary algorithms. In: Guervós, J.M., Adamidis, P., Beyer, H.G., Fernández-Villacañas, J. L., Schwefel, H.P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 665–675. Springer, Heidelberg (2002)
3. Eiben, A.E., Schoenauer, M., van Krevelen, D.W.F., Hobbelman, M.C., ten Hagen, M.A., van het Schip, R.C.: Autonomous selection in evolutionary algorithms. In: GECCO 2007, pp. 1506–1506. ACM Press, New York (2007)
4. Giacobini, M., Preuss, M., Tomassini, M.: Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In: Gottlieb, J., Raidl, G.R. (eds.) EvoCOP 2006. LNCS, vol. 3906, pp. 85–96. Springer, Heidelberg (2006)
5. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23(3), 219–252 (2005)
6. Jelasity, M., van Steen, M.: Large-scale newscast computing on the Internet. Technical Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands October (2002)
7. Jesi, G.P.: Peersim, a peer-to-peer simulator, `http://peersim.sourceforge.net/`
8. Laredo, J.L.J., Eiben, E.A., Schoenauer, M., Castillo, P.A., Mora, A.M., Merelo, J.J.: Exploring selection mechanisms for an agent-based distributed evolutionary algorithm. In: GECCO 2007, pp. 2801–2808. ACM Press, New York (2007)

9. Laredo, J.L.J., Valdivieso, P.A.C., Paechter, B., Mora, A.M., Alfaro-Cid, E., Esparcia-Alcázar, A., Guervós, J.J.M.: Empirical validation of a gossiping communication mechanism for parallel EAs. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 129–136. Springer, Heidelberg (2007)
10. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore (2005)
11. Tomassini, M.: Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series). Springer, New York (2005)
12. Wickramasinghe, W.R.M.U.K., van Steen, M., Eiben, A.E.: Peer-to-Peer evolutionary algorithms with adaptive autonomous selection. In: GECCO 2007, pp. 1460–1467. ACM Press, New York (2007)

Mixed-Integer Evolution Strategies with Dynamic Niching

Rui Li[1], Jeroen Eggermont[2], Ofer M. Shir[1], Michael T.M. Emmerich[1], Thomas Bäck[1], Jouke Dijkstra[2], and Johan H.C. Reiber[2]

[1] Natural Computing Group, Leiden University, P.O. Box 9500, 2300 CA Leiden, The Netherlands
{ruili,oshir,emmerich,baeck}@liacs.nl

[2] Division of Image Processing, Department of Radiology C2S, Leiden University Medical Center, P.O. Box 9600, 2300 RC Leiden, The Netherlands
{J.Eggermont,J.Dijkstra,J.H.C.Reiber}@lumc.nl

Abstract. Mixed-Integer Evolution Strategies (MIES) are a natural extension of standard Evolution Strategies (ES) for addressing optimization of various types of variables – continuous, ordinal integer, and nominal discrete – at the same time. Like most Evolutionary Algorithms (EAs), they experience problems in obtaining the global optimum in highly multimodal search landscapes. Niching methods, the extension of EAs to multimodal domains, are designed to treat this issue. In this study we present a dynamic niching technique for Mixed-Integer Evolution Strategies, based upon an existing ES niching approach, which was developed recently and successfully applied to continuous landscapes. The new approach is based on the heterogeneous distance measure that addresses search space similarity in a way consistent with the mutation operators of the MIES. We apply the proposed Dynamic Niching MIES framework to a test-bed of artificial landscapes and show the improvement on the global convergence in comparison to the standard MIES algorithm.

1 Introduction

Evolutionary Algorithms (EAs) have the tendency to converge to a single solution [2,19], even if the search landscape has multiple globally optimal solutions. This is due to effects such as genetic drift [12] , fast takeover [2], and disruptive recombination [10]. Population diversity loss in EAs does not only make it difficult to obtain multiple global optima, but may also prevent the algorithm from locating the global optimum.

Niching techniques have been proposed to counteract population diversity loss in EAs. They support parallel convergence into multiple attraction basins in a multimodal landscape within a single run. Niching techniques have been mainly developed within the framework of Genetic Algorithms (GAs) in the past decades (see, e.g. [16] and [19]), and have recently also received increasing attention from the Evolution Strategies (ES) community [10,15,17,18].

The application of niching in ES proved to be very successful in improving convergence reliability and solution diversity in multimodal continuous optimization. However, it remains an open question, whether niching also can be incorporated into mixed-integer search spaces, which are of great practical relevance [3]. In this paper we investigate, whether niching is also beneficial in this problem domain by combining the niching approach by Shir et al. [15] with the Mixed-Integer Evolution Strategy (MIES) [4,7].

A crucial step will be the definition of an appropriate metric that is compatible with the neighborhood structures used by the search operators of the Mixed-Integer Evolution Strategies. Thereby we aim for a coherent algorithm design which will make a theoretical analysis of the algorithm more accessible. It is a known drawback that the MIES has difficulties to converge to global optima of highly multimodal landscapes [6]. Based on selected test problems, such as Mixed-Integer NK Landscapes [6] and Barrier Functions [7], we study whether the introduction of niching improves the MIES performance on such landscapes.

The paper is structured as follows: In Section 2 we review the Dynamic Niching ES for continuous multimodal optimization. Then, in Section 3, the Mixed-Integer Evolution Strategy is described. A combination of dynamic niching and Mixed-Integer Evolution Strategies using heterogenous distance measures will be proposed in Section 4. Experiments on multimodal mixed-integer landscapes will be reported and discussed in Section 5. Finally, in Section 6, we summarize conclusions and discuss open questions for future research.

2 Dynamic Niching Evolution Strategies

Next, we outline and discuss the Dynamic Niching ES Algorithm [14] in detail. The algorithm starts with the initialization of q niches with μ individuals and their evaluation. Then, the following loop is repeated until a termination criterion is met: Firstly, for each niche the algorithm generates λ offspring based on the μ parents. Depending on the instantiation of the algorithmic ES kernel, mutation and recombination operators are employed for this purpose.

By restricting recombination to the dynamically updated niches, the algorithm enforces a mating restriction scheme which allows competitive mating only within the niches. This is done to prevent disruptive effects of the recombination operator [10]. The concept of fixed mating resources is strictly enforced: For every niche the same number of offspring is generated, also referred to as the *niche hosting capacity*. This measure is taken in order to prevent genetic drift effects, as described e.g. in [12].

Upon the fitness evaluation of the new individuals, offspring and parent individuals are merged into one population comprising now $q \times (\mu + \lambda)$ individuals. The algorithm then employs a sub-routine for dynamically identifying the various fitness-peaks of every generation (which uniquely define the niches) and then assigns each individual to a niche. The classification into niches is carried out in a *greedy* manner, by means of the so-called Dynamic Peak Identification (DPI) algorithm [9]. The latter is outlined as Algorithm 1.

Besides the global selection phase taking place in the niche forming process, which will be described later, a local environmental selection takes place within each niche, that enables step-size adaptation to the local topography of the niches. If the number of individuals in a peak set is less than μ, the algorithm creates new samples in the search space and adds them to the niche until it contains μ individuals. A summary of the algorithm is given in Algorithm 2.

Algorithm 1. Dynamic Peak Identification (DPI)

```
in: population $Pop$, # niches $q$, niche radius $\rho$, out: peak sets $DPS$
 1: Sort $Pop$ in decreasing fitness order
 2: $i := 1$
 3: $NumPeaks := 0$
 4: $DPS := \emptyset$ {Set of peak elements in population}
 5: while $NumPeaks \neq q$ and $i \leq popSize$ do
 6:   if $Pop[i]$ is not within sphere of radius $\rho$ around peak in $DPS$ then
 7:     $DPS := DPS \cup \{Pop[i]\}$
 8:     $NumPeaks := NumPeaks + 1$
 9:   end if
10:   $i := i + 1$
11: end while
```

The number of expected niches, q, is given as input to the algorithm. The distance calculation is implemented with the Euclidean metric in the decision parameter space since all parameters are continuous. The niche radius ρ itself is approximated a-priori with Eq. 1, and remains fixed during the run.

$$\rho = \frac{r}{\sqrt[n]{q}} \quad \text{with} \quad r = \frac{1}{2}\sqrt{\sum_{k=1}^{n}(x_{k,max} - x_{k,min})^2} \tag{1}$$

with $x_{k,min}$ and $x_{k,max}$ the lower and upper boundary values of parameter x_k.

3 Mixed-Integer Evolution Strategies

Mixed-Integer Evolution Strategies (MIES) are a special variant of ES, introduced in [4], designed to tackle so-called mixed integer optimization problems (MIOP). In MIOP different types of discrete and continuous optimization variables occur in combination. MIES deal with three variable types:

Continuous Variables. Floating point numbers the value of which can be adjusted gradually within a range.

Integer Variables. Parameter values that have one or two nearest neighbors and thus a minimal variation can be defined.

Nominal Discrete Variables. There is no metric or ordering defined on the finite domain of this discrete variable.

Algorithm 2. Niching-ES.

```
in: Number of niches $q$, Niche radius $\rho$, out: optimized solution(s)
 1: Initialize $q$ equally-sized niches of size $\mu$ randomly
 2: Evaluate all new individuals in all niches
 3: while Termination criteria not full filled do
 4:    for every niche $i = 1 \ldots q$ do
 5:       generate $\lambda$ offspring from $\mu$ parents
 6:       Evaluate fitness of $\lambda$ offspring individuals
 7:       Update best found solution(s)
 8:    end for
 9:    Combine all $\mu + \lambda$ individuals from niches into one population
10:    Compute the Dynamic Peak Set with DPI (Algo. 1)
11:    Select $\mu$ best individuals per niche
12:    for every niche $i = 1 \ldots q$ do
13:       if $\mu_i$ = number of individuals in niche i $< \mu$ then
14:          Generate and Evaluate $\mu - \mu_i$ new individuals
15:       end if
16:    end for
17: end while
```

These variable types differ in two aspects: (1) The cardinality of their domain: While the continuous domain contains over-countable many solution, the domain of the integer parameters is either countable or finite, depending on the chosen range, and the domain for the nominal parameters is finite. (2) The metrics that are used to describe similarity of solution vectors. Unless further knowledge of the problem is available, three distance measures seem appropriate for measuring similarity between solution vectors in a straightforward manner: In the continuous domain this is the Euclidean distance, for the integer domain the Manhattan distance (i.e. the sum of absolute vector differences). For the nominal discrete parameters the Hamming distance or overlap distance that counts the number of positions in which two tuples of nominal discrete variables differ may serve as a straightforward choice of a metric.

In order to deal with these different variable types Mixed-Integer Evolution Strategies use specialized operators. In contrast to the mutation operator in standard evolution strategies, the mutation procedure for mixed-integer spaces works with a heterogeneous distribution when sampling an offspring solution based on a parent solution, combining the ℓ_2 symmetric normal distribution for continuous variables [13], the ℓ_1 symmetric difference of two geometric distributions for integer variables [11], and a uniform distribution for the nominal discrete values [2]. To scale mutation strength and enable self-adaptation, step size variables are introduced for continuous and integer variables, and mutation probabilities for the nominal discrete variables. For a more detailed description of the MIES we refer the reader to [7].

4 Dynamic Niching for Mixed-Integer ES

To incorporate MIES into the Dynamic Niching ES framework we must define a proper distance metric for the mixed-integer space. For continuous spaces the Euclidean metric seems to be a straightforward choice, while for nominal discrete spaces an overlap metric seems suitable, as it does not assume any continuity of the objective function w.r.t. a particular ordering of the domain. For two integer parameter vectors the distance can be measured by means of the Manhattan distance in a straightforward way. This is the accumulated distance when computing the difference of single parameter values of the variables. In combination with the MIES the choice of the Manhattan distance is also in conformity with the symmetry assumptions used in the design of the mutation operator, which generates samples from an ℓ_1 symmetric distribution. We combine the different metrics using the Heterogeneous Euclidean-Overlap Metric (HEOM) approach by Wilson and Martinez [20].

Let $\Delta_r(\mathbf{r}, \mathbf{r}') = \sum_{i=1}^{n_r} (r_i - r_i')^2$, $\Delta_z(\mathbf{z}) = \sum_{i=1}^{n_z} |z_i - z_i'|$, and $\Delta_d(\mathbf{d}, \mathbf{d}') = \sum_{i=1}^{n_d} I(d_i \neq d_i')$ with $I(true) = 1, I(false) = 0$.

Then the combined heterogeneous metric Δ_h for $\mathbf{h} = (\mathbf{r} \circ \mathbf{z} \circ \mathbf{d})$ reads:

$$\Delta_h(\mathbf{h}, \mathbf{h}') = \sqrt{\Delta_r(\mathbf{r}, \mathbf{r}') + \Delta_z(\mathbf{z}, \mathbf{z}') + \Delta_d(\mathbf{d}, \mathbf{d}')}. \tag{2}$$

By using the aforementioned heterogeneous metric, the niche radius ρ in mixed-integer search space now can be approximated as follows:

$$\rho = \frac{r}{\sqrt[n]{q}} \quad \text{with} \quad r = \frac{1}{2}\sqrt{\sum_{k=1}^{n} \Delta_x(x_{k,max}, x_{k,min})} \tag{3}$$

Here $x_{k,min}$ and $x_{k,max}$ denote the lower and upper boundary values of parameter x_k and q denotes the number of peaks in the solution space. We assumed that every niche with radius ρ occupies $\frac{1}{q}$-th of the entire volume of the space.

5 Test Functions and Experimental Results

To investigate the behavior of our algorithm, we applied it to two carefully designed mixed-integer multimodal functions in various dimensions. Specifically, we are interested in the global convergence. Performance comparison between Dynamic Niching MIES with standard MIES is also presented.

5.1 Barrier Functions

Barrier functions, introduced in [7], create mixed-integer optimization problems with a scalable degree of ruggedness (determined by parameter C) by generating an integer array `A` using Algorithm 3.

Algorithm 3. Algorithm to generate array A for the Barrier function.

```
$\mathtt{A}[i] = i, i = 0, \ldots, 20$
for $k \in \{1, \ldots, C\}$ do
    $j \leftarrow$ random number out of $\{0, \ldots, 19\}$
    swap values of $\mathtt{A}[j]$ and $\mathtt{A}[j+1]$
end for
```

The barrier function f_{barrier} is computed as:

$$f_{\text{barrier}}(\mathbf{r}, \mathbf{z}, \mathbf{d}) = \sum_{i=1}^{n_r} \mathtt{A_i}[\lfloor r_i \rfloor]^2 + \sum_{i=1}^{n_z} \mathtt{A_i}[z_i]^2 + \sum_{i=1}^{n_d} \mathtt{B_i}[d_i]^2 \rightarrow \min \tag{4}$$

$$n_r = n_z = n_d = 5, \mathbf{r} \in [0, 20]^{n_r} \subset \mathbb{R}^{n_r},$$
$$\mathbf{z} \in [0, 19]^{n_z}, \mathbf{d} \in \{0, \ldots, 19\}^{n_d}$$

Here, $B_i[0], \ldots, B_i[19]$ denotes a set of i permutations of the sequence $0, \ldots, 19$, each of which is chosen randomly before the run. This is done to prevent the nominal value d_i from being quantitatively (anti-)correlated with the value of the objective function f_{barrier} which would contradict with the assumption that d_i are nominal values.

The ruggedness of the resulting barrier function with regard to the integer space is controlled by parameter C with higher values of C resulting in more rugged landscapes with many barriers. To illustrate the influence of C on the geometry of the function we included plots for two-variable instantiations of the barrier function in Figure 1.

To test the Dynamic Niching MIES and standard MIES algorithm we generated barrier functions for $C = 20$, $C = 200$, $C = 2000$ and $C = 5000$ and ran both the Dynamic Niching MIES and a standard MIES algorithms 20 times with different random seeds. For the Dynamic Niching MIES we used 5 niches with $\mu = 15$ and $\lambda = 75$ for each niche. For the MIES algorithm we used a $(75 + 500)$ strategy thereby making sure that the number of parents, offspring and fitness evaluations per generation is the same for both algorithms.

The results of the experiments are displayed in Figure 2. Although the Dynamic Niching MIES converges a little slower than the standard MIES algorithm it does reach the same performance in the end. In the case of C=2000 Dynamic Niching MIES performs slightly better than the standard MIES on average. The possible explanation is that the barrier function landscape with C=2000 is harder than others. The standard MIES converges faster but Dynamic Niching MIES has a better chance of getting rid of local traps at last.

5.2 Mixed-Integer NK Landscapes

NK landscapes (NKL, also referred to as NK fitness landscapes), introduced by Kauffman [5], were devised to explore the way that epistasis controls the 'ruggedness' of an adaptive landscape. They are particularly used as test problem generators for Genetic Algorithms (GAs) to understand the dynamics of evolutionary search. The ruggedness and the degree of interaction between variables of NKL can be easily controlled by two tunable parameters: the number of genes N and the number of epistatic links of each gene to other genes K. Moreover, for given values of N and K, a large number of NK landscapes can be created at random.

Mixed-Integer NK-Landscapes (MI-NKL) were introduced in [6] and are an extension of NKL from the traditional binary case to a mixed variable case with continuous, nominal discrete, and integer variables. The resulting test function generator is a suitable test model for our dynamic niching Mixed-Integer Evolution Strategy.

In order to test our Dynamic Niching MIES algorithm we test it on different Mixed-Integer NK landscapes with 15 variables (5 continuous (range $[-10, 10]$), 5 integer variables (also range $[-10, 10]$) and 5 nominal discrete variables (Boolean ($\{0, 1\}$)). We generated 10 random MI-NKL for different levels of K (2, 5, 10, and 14) to simulate different problem difficulties and both the Dynamic Niching MIES and standard MIES algorithms were run 20 times on each MI-NKL using different random seeds. We used a total population size of 75 for both the standard MIES and Dynamic Niching MIES algorithm (15 individuals per niche) and an offspring size of 500 (100 per niche). To compare (and average) the results of the different experiments we used the following error-measure:

$$\text{error} = \text{best found fitness - best possible fitness}$$

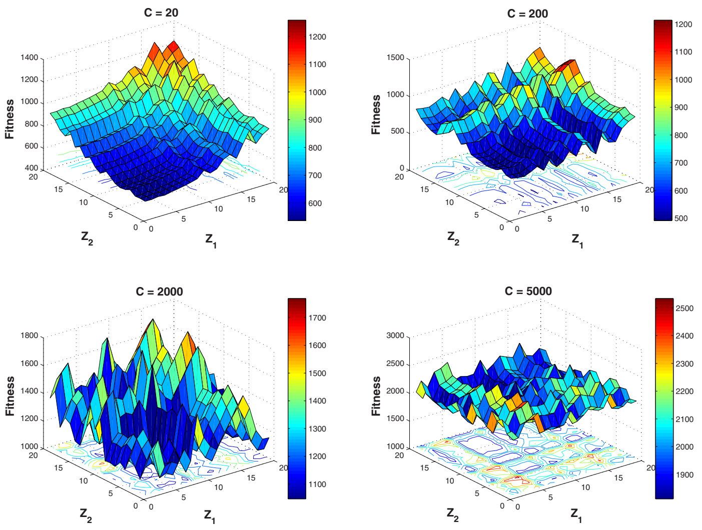


Fig. 1. 3-D shaded surface plot with contour plot of the barrier test function for two integer variables Z_1 and Z_2, the control parameter C = 20, 200, 2000 and 5000. All other variables were kept constant at a value of zero, Z_1 and Z_2 values were varied in the range from 0 to 19.

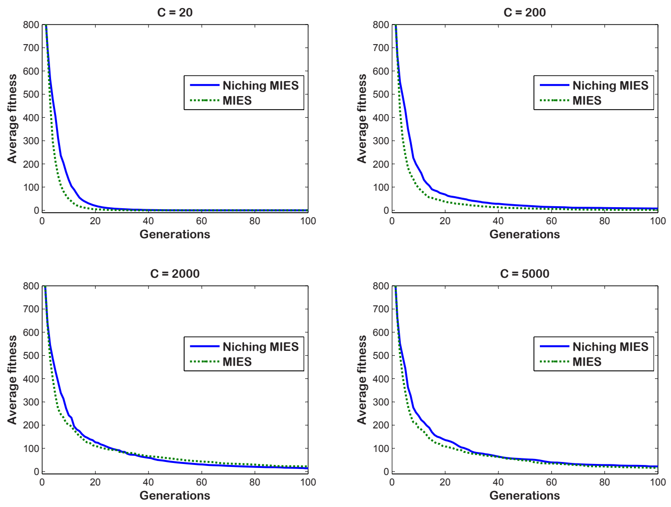


Fig. 2. Average best fitness results over 20 experiments for barrier functions with $C = 20$, $C = 200$, $C = 2000$ and $C = 5000$ for both the Dynamic Niching MIES and standard MIES algorithms

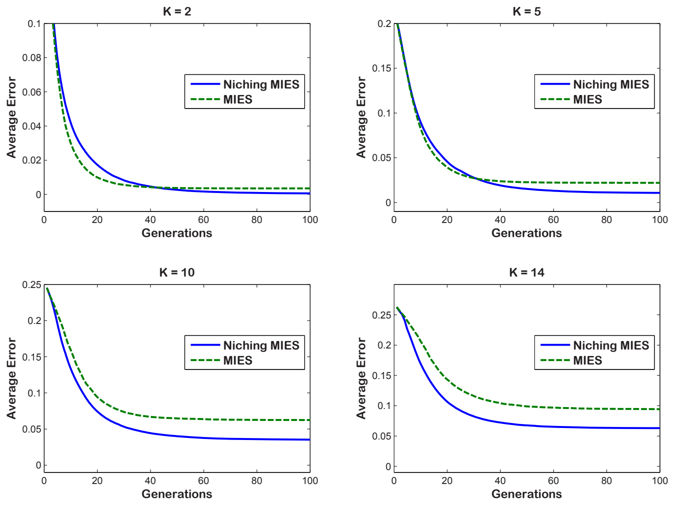


Fig. 3. The error average of both Dynamic Niching MIES and standard MIES on different mixed-integer NK landscape problems with N = 15

The results of the experiments are displayed in Figure 3. For $K = 2$ and $K = 5$ we see, similar to the results of the barrier functions, that the standard MIES algorithm converges faster. However, on the MI-NKL the Dynamic Niching

MIES algorithm manages to achieve a better result on average. If we look at the results for more rugged (and harder) MI-NKL with $K = 10$ and $K = 14$ we see that the Dynamic Niching MIES outperforms the standard MIES algorithm both in convergence speed and final solution quality.

We also compared the number of experiments the Dynamic Niching MIES and MIES algorithms find the global optimum. For $K = 2$ the Dynamic Niching MIES algorithm finds the optimum 174 times out of 200 (10 different MI-NKL times 20 runs) while MIES finds it 143 times. As K increases both algorithms find the optimum less often, which is expected since the difficulty increases. For $K = 5$, 10 and 14 the Dynamic Niching MIES finds the optimum 92, 19 and 8 times respectively. MIES only manages to finds the optimum 67, 6 and 3 times for $K = 5$, 10 and 14. Thus, the Dynamic Niching MIES algorithm does not only result in a lower average error but also manages to find the global optimum more often.

6 Conclusions and Outlook

Studies on artificial landscapes reveal that the proposed heterogeneous niching can be a useful ingredient in highly rugged landscapes. On MI-NK Landscapes it clearly improves the chances to obtain the global optimum. In more simple landscapes it only slightly slows down the convergence speed compared with standard MIES. In conclusion, it can be said that in case of simple problems the usage of the new strategy will not be harmful and in the case of highly rugged problems it can lead to solutions of better quality than standard MIES.

In the future the Dynamic Niching MIES should be tested on additional problems, including real-world applications. Moreover, a deepened understanding of niche formation process in mixed-integer landscapes and the influence of strategy parameters may help to further improve its performance.

Acknowledgements. This research is supported by the Netherlands Organisation for Scientific Research (NWO).

References

1. Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, Edinburgh, UK, September 2-4, 2005. IEEE, Los Alamitos (2005)
2. Bäck, Th.: Evolutionary algorithms in theory and practice. Oxford University Press, New York (1996)
3. Bäck, Th., Schütz, M.: Evolution strategies for mixed-integer optimization of optical multilayer systems. In: Evolutionary Programming, pp. 33–51 (1995)
4. Emmerich, M., Grötzner, M., Groß, B., Schütz, M.: Mixed-integer evolution strategy for chemical plant optimization with simulators. In: Parmee, I.C. (ed.) Evolutionary Design and Manufacture - Selected papers from ACDM 2000, pp. 55–67. Springer, Heidelberg (2000)
5. Kauffman, S.: Towards a general theory of adaptive walks on rugged landscapes. Journal of theoretical biology 128(1), 11–45 (1987)

6. Li, R., Emmerich, M.T.M., Eggermont, J., Bovenkamp, E.G.P., Bäck, Th., Dijkstra, J., Reiber, J.H.C.: Mixed-Integer NK Landscapes. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 42–51. Springer, Heidelberg (2006)
7. Li, R., Emmerich, M.T.M., Eggermont, J., Bovenkamp, E.G.P., Bäck, Th., Dijkstra, J., Reiber, J.H.C.: Mixed-integer optimization of coronary vessel image analysis using evolution strategies. In: Cattolico, M. (ed.) [8], pp. 1645–1652
8. Cattolico, M. (ed.): Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2006. ACM Press, Seattle (2006)
9. Miller, B.L., Shaw, M.J.: Genetic algorithms with dynamic niche sharing for multimodal function optimization. In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC 1996), pp. 786–791 (1996)
10. Preuss, M., Schönemann, L., Emmerich, M.T.M.: Counteracting genetic drift and disruptive recombination in $(\mu + /, \lambda)$-ea on multimodal fitness landscapes. In: Beyer, H.-G., O'Reilly, U.-M. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2005, pp. 865–872. ACM Press, New York (2005)
11. Rudolph, G.: An evolutionary algorithm for integer programming. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 139–148. Springer, Heidelberg (1994)
12. Schönemann, L., Emmerich, M.T.M., Preuss, M.: On the extinction of evolutionary algorithms sub-populations on multimodal landscapes. Informatica - Special Issue on Bioinspired Optimization 28(4), 345–351 (2004)
13. Schwefel, H.-P.: Evolution and Optimum Seeking: The Sixth Generation. John Wiley & Sons, Inc., New York (1993)
14. Shir, O.M., Bäck, Th.: Dynamic niching in evolution strategies with covariance matrix adaptation. In: Congress on Evolutionary Computation [1], pp. 2584–2591
15. Shir, O.M., Bäck, Th.: Niching with Derandomized Evolution Strategies in Artificial and Real-World Landscapes. Natural Computing (2008)
16. Singh, G., Deb, K.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Cattolico, M. (ed.) [8], pp. 1305–1312
17. Stoean, C., Preuss, M., Gorunescu, R., Dumitrescu, D.: Elitist generational genetic chromodynamics - a new radii-based evolutionary algorithm for multimodal optimization. In: Congress on Evolutionary Computation [1], pp. 1839–1846
18. Streichert, F., Stein, G., Ulmer, H., Zell, A.: A clustering based niching method for evolutionary algorithms. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 644–645. Springer, Heidelberg (2003)
19. Mahfoud, S.W.: Niching Methods for Genetic Algorithms. PhD thesis, University of Illinois at Urbana Champaign (1995)
20. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. Journal of Artificial Intelligence Research 6, 1–34 (1997)

A Compass to Guide Genetic Algorithms

Jorge Maturana and Frédéric Saubion

LERIA, Université d'Angers
2, Bd Lavoisier 49045 Angers, France
{maturana,saubion}@info.univ-angers.fr

Abstract. Parameter control is a key issue to enhance performances of Genetic Algorithms (GA). Although many studies exist on this problem, it is rarely addressed in a general way. Consequently, in practice, parameters are often adjusted manually. Some generic approaches have been experimented by looking at the recent improvements provided by the operators. In this paper, we extend this approach by including operators' effect over population diversity and computation time. Our controller, named Compass, provides an abstraction of GA's parameters that allows the user to directly adjust the balance between exploration and exploitation of the search space. The approach is then experimented on the resolution of a classic combinatorial problem (SAT).

1 Introduction

Genetic Algorithms (GA) are metaheuristics inspired by natural evolution, which manage a population of individuals that evolve thanks to operators' applications. Since their introduction, GAs have been successfully applied to solve various complex optimization problems. From a general point of view, the performance of a GA is related to its ability to correctly explore and exploit the interesting areas of the search space. Several parameters are commonly used to adjust this exploration/exploitation balance (EEB), and the operator application rates are probably among the most influential ones. A suitable control of parameters is crucial to avoid two well-known problems: premature convergence, that occurs when the population gets trapped in a local optima, and the loss of computation time, due to the inability of the GA to detect the most promising areas of the search space. Most of the efforts on this subject are only applicable to specific algorithms, thus, in practice, parameter control is often achieved manually, supported by empirical observations. More recently, new methods have begun to rise up, proposing more generic control mechanisms. In this trend, our motivation is to design a new controller in which parameters could be handled by more general and abstract concepts, in order to be used by a wide range of GAs.

Techniques for assigning values to parameters can be classified according to the taxonomy proposed by Eiben et al. [1]. A general class, named *Parameter Setting* [2], is divided in *Parameter Tuning*, where parameters are fixed before the run, and *Parameter Control*, where parameters are modified during the run. Parameter Control is further divided in *Deterministic*, where parameters are modified according to a fixed and predefined scheduling; *Adaptive*, where the current state of the search is used to modify parameters by means of rules; and *Self-Adaptive* [3], where parameters are encoded in the genotype and evolve together with the population.

Within adaptive control, the central issue is to design rules able to guide the search and to make the suitable choices. A straightforward way consists in performing test runs to extract pertinent information in order to feed the system. However, this approach involves an extra computational time and does not really correspond to the idea of an "automatic self-driven" algorithm.

A more sophisticated way to build a control system consists in adding a learning component, which is able to identify a correct control procedure. This reduces prior effort and increases adaptation abilities, according to the needs of different algorithms. In this context, two perspectives could be identified:

The first approach consists in modeling the behavior of the GA using different parameters, typically during a learning phase. [4] presents two methods including a learning phase that tries different combinations of parameters and encodes the results in tables or rules. A similar approach is presented in [5], where population's diversity and fitness evaluation are embedded in fuzzy logic controllers. Later this controllers are used to guide the search according to a high level strategy. [6] proposes an algorithm divided in periods of learning and control of parameters, by adjusting central and limit values of them.

A second approach consists in providing a fast control, neglecting the modeling aspect. [7] presents a controller that adjusts operators' rates according to recent performances. Similar ideas are presented in [8,9]. In [10], this approach is extended by considering several statistics of individuals fitness and survival rate to evaluate operator quality. In [11], the population is resized, depending of several criteria based on the improvement of the best historical fitness. [12] presents an algorithm that oscillates between exploration and exploitation phases when diversity thresholds are crossed. [13] modifies parameters according to best fitness value. Some methods in this class require special features from the GA, such as [14], that maintains several populations with different parameter values, and moves the parameter's values toward the value that produces the best results. In [15], a forking scheme is used: a parent population is in charge of exploration, while several child populations exploit particular areas of the search space. In [16], a parameterless GA gets rid of *popsize* parameter by comparing the performance of multiple populations of different size.

In this paper, we investigate a combination of these two general approaches in order to benefit from their complementary strengths, providing an original abstract control of GAs' operators. Our controller measures the variations of population's diversity and mean fitness resulting from an operator application, as well as its execution time. A unique control parameter (Θ) allows us to adjust the desired level of EEB and determines the application rates assigned to each operator. We have tested our approach on the resolution of the famous boolean satisfaction problem (SAT) and compared it to other adaptive control methods.

The paper is organized as follows. Sect. 2 exposes our approach, Sect. 3 describes the experimental framework we have used, and Sect. 4 discusses results. Finally, main conclusions and future directions are drawn in Sect. 5.

2 Method Overview

We consider here a basic steady-state GA: at each step an operator is selected among several ones, according to a variable probability. Asexual operators are applied to the best of two randomly chosen individuals of the population, and the resulting individual replaces the worst one. Sexual operators work on two randomly chosen individuals, modifying them directly. The parameters considered here are therefore operators' application rates.

As mentioned in the introduction, adaptive control can be considered from two different points of view. In order to illustrate more precisely these differences, we may detail two recent and representative approaches by comparing the work of Thierens [7] and a method proposed by Wong et al. [6].

In [7], Adaptive Pursuit (AP) aims at adjusting the probabilities of associated operators, depending on their performances, measured typically by fitness improvement during previous applications. This method is able to quickly adapt these probabilities in order to award the most successful operators. AP does not care about understanding the behavior of the algorithm and focuses immediately on the best values, in order to increase the performance. At this point, we may remark that algorithms that are solely based on fitness improvement may experience premature convergence.

In the APGAIN method [6], the search is divided in epochs, further divided in two periods. The first one is devoted to the measurement of operators' performance by applying them randomly, and the second one applies operators according to a probability which is proportional to the observed performance. Three values (low, medium, high) are considered for each parameter, and adjusted by moving them towards the most successful value. Finally, a diversification mechanism is included in the fitness function. Roughly a quarter of the generations is dedicated to the first (learning) period, what could be harmful if there are disrupting operators.

Here, we propose a new controller (Compass) based on the idea presented in [5], that considers both diversity and quality as pertinent criteria to evaluate algorithms' performance. Parameters are abstracted, in order to guide the search by inducing a required level of EEB. The operators are evaluated after each application and, in addition to diversity and fitness variation, a third measure –operator's execution time– is also considered. To get rid of previous drawbacks, we include some controllers' features that adapt parameters' rates during the search [7,8,9], namely the speed of response, to update the model. Operators are applied according to their application rate, which is updated at every generation. Since we are interested in a controller which could be used by any GA, it must be independent and placed at a different layer. We have then implemented Compass in a C++ class, included by the GA.

2.1 Operator Evaluation and Applications Rates Updating

Given an operator $i \in [1 \dots k]$ and a generation number t, let d_{it}, q_{it}, T_{it} be, respectively, the population's mean diversity variation, mean quality (fitness) variation, and mean execution time of i over the last τ applications of this operator. At the beginning of the run, all operators can be applied with the same probability.

We define a vector $o_{it} = (d_{it}, q_{it})$ to characterize the effects of the operator over the population in terms of variation of quality and diversity (axis $\varDelta D$ and $\varDelta Q$ of Fig. 1). Note that, since both quality and diversity improvements correspond to somewhat opposite goals, most vectors will lay on quadrants II (improvement of quality but a decrease in diversity) and IV (increase in diversity and a reduction of mean fitness), shown in Fig. 1a.

Algorithms that just consider the fitness improvement to adjust the operator probabilities would only use the projection of o_{it} over the y-axis (dotted lines in Fig. 1b). On the other hand, if diversity is solely taken in account, measures would be considered as the projection over the x-axis (Fig. 1c).

Our goal is to control these two criteria together by choosing a search direction which will be expressed by a vector c (defined by its angle $\varTheta \in [0, \frac{\pi}{2}]$) that characterizes also its orthogonal plane P (see Fig. 1d).

Since measures of diversity and quality usually have different magnitudes, they are normalized as:

$$d_{it}^n = \frac{d_{it}}{max_i\{|d_{it}|\}} \qquad \text{and} \qquad q_{it}^n = \frac{q_{it}}{max_i\{|q_{it}|\}}$$

We thus have vectors $o_{it}^n = (d_{it}^n, q_{it}^n)$. Rewards are then based on the projection of vectors o_{it}^n over c, i.e., $|o_{it}|cos(\alpha_{it})$, α_{it} being the angle between o_{it} and c. A value of $\varTheta$ close to 0 will encourage exploration, while a value close to $\frac{\pi}{2}$ will favor exploitation. In this way, the management of application rates is abstracted by the angle $\varTheta$, that guides the direction of the search as the needle of a compass shows the north.

Projections are turned into positive values by subtracting the smallest one and dividing them by execution time, in order to award faster operators (Fig. 1e).

$$\delta_{it} = \frac{|o_{it}^n|cos(\alpha_{it}) - min_i\{|o_{it}^n|cos(\alpha_{it})\}}{T_{it}}$$

Application rates are obtained proportionally to values of δ_{it} plus a constant ξ_t, that ensures that the smallest rate is equal to a minimal rate, P_{min}, preventing the disappearance of the corresponding operator (Fig. 1f).

$$p_{it} = \frac{\delta_{it} + \xi_t}{\sum_{i=1}^{k} \delta_{it} + \xi_t}$$

2.2 Operator Application

Operators' application rates are updated at every generation. An interesting phenomenon, observed during previous experiments, is the displacement of points

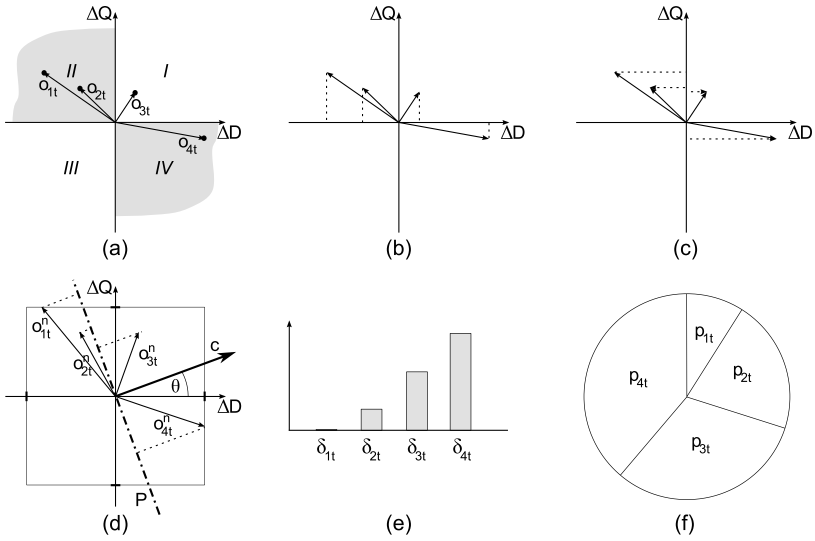


Fig. 1. (a) points (d_{it}, q_{it}) and corresponding vectors o_{it}, (b) quality-based ranking, (c) diversity-based ranking, (d) proposed approach, (e) values of δ_{it}, (f) final probabilities

(d^n_{it}, q^n_{it}) in the graphic during execution. Consider for instance that Θ is set to $\frac{\pi}{4}$, so an equal importance is given to ΔQ and ΔD. At the beginning of the search, just after the population was randomly created, the population diversity is high and mean fitness is low, thus it is easy for most operators to be situated in the quadrant II. After some generations, the population starts to converge to some optimum, so improvement becomes difficult, and points in II corresponding to exploitative operators move near x-axis. When improvements in this zone are exhausted, exploitation operators obtain worst rewards than exploration ones, causing a shift of the search to diversification, and escaping from that optimum. Such a visualization tool could be useful to understand the behavior of operators as well as for debugging purposes.

3 Experimentation

For our experiments, we focus on the use of GAs for the resolution of combinatorial problems. Among the numerous possible classes, we have chosen the Boolean satisfiability problem (SAT) [17], which consists in assigning values to binary variables in order to satisfy a Boolean formula.

The first reason is that this is probably the most known combinatorial problem, since it has been the first to be proved NP complete and therefore it has been used to encode and solve problems from many application areas. The second reason is that there exists an impressive library [18] of instances and their difficulty has been deeply studied with several interesting theoretical results (e.g., phase transition), which allows us to select different instances with various search landscapes' properties.

More formally, an instance of the SAT problem is defined by a set of Boolean variables $\mathcal{X} = \{x_1, ..., x_n\}$ and a Boolean formula $\mathcal{F}\colon \{0,1\}^n \rightarrow \{0,1\}$. The formula is said to be satisfiable if there exists an assignment $v\colon \mathcal{X} \rightarrow \{0,1\}^n$ satisfying $\mathcal{F}$ and unsatisfiable otherwise. Instances are classically formulated in conjunctive normal form (conjunctions of clauses) and therefore one has to satisfy all these clauses.

To solve this problem, we consider a GA with a binary population that applies one operator at each generation. The fitness function evaluates the number of clauses satisfied by an individual and the associated problem is thus obviously a maximization one. The diversity is classically computed as the Hamming distance entropy (see [19]).

In order to evaluate our control approach, we compare it with Adaptive Pursuit (AP) [7] and APGAIN [6]. As mentioned in Sect. 2, AP is representative of many controllers that consider fitness improvement as their guiding criterion while APGAIN is representative of methods that try to learn and model the behavior of the operators. Additionally, we also included a uniform choice (UC) among operators as the baseline of the comparison. In order to check the robustness of our method –but restricted by the lack of space–, we present 13 different instances from the SATLIB repository [18], mixing problems of different sizes and nature, including random-generated instances, graph coloring, logistics planning and blocks world problems.

3.1 Operators

The goal of this work is to create an abstraction of operators, regardless of their quality, and to compare controllers, and not to develop an efficient GA for SAT. The idea is also to use non standard operators, whose effect over diversity and quality is a priori unknown. Therefore, we propose six operators with different features, more or less specialized with regards to the SAT problem.

One-point crossover chooses randomly two individuals and crosses them at a random position. In this operator exclusively, the best child replaces the worst parent.

Contagion chooses randomly two individuals, and the variables in false clauses of the worst one are replaced with corresponding values of the best individual.

Hill climbing checks all neighbors by swapping one variable, moves to the better one and repeats while improvement is possible.

Tunneling swaps variables without decreasing the number of true clauses according to a tabu list of length equal to $\frac{1}{4}$ of the number of variables.

Badswap swaps all variables that appear in false clauses.

Wave chooses the variable that appears in the highest number of false clauses and in the minimum number of clauses only supported by it, and swaps it. It repeats the same process at most $\frac{1}{2}$ times the number of variables.

In order to observe the effect of population size over the performance of controllers, we performed experiments with populations of 3, 5, 10 and 20 individuals. 10.000 generations were processed, in order to observe the long-term behavior of controllers.

3.2 Control Strategy

Previous experiments have shown that values of Θ around 0.25π produced good results. To observe the sensitivity of this value, we ran experiments with values of 0.20π, 0.25π and 0.30π. Note that, even when the value of Θ remains fixed along the run, it does not mean that Compass falls in the category of parameter tuning. It is necessary to distinguish the parameters of the GA (operator's application rates) from the parameter(s) of the control strategy (θ in this case). Controller parameters provide an abstraction of GA's parameters. It is pertinent to wonder whether it is worth replacing GA parameters by controller parameters. We think that this substitution is beneficial in two cases:

- When the effect of controller parameters is less sensitive than GA's parameters. Consider, for instance, the case of mutation rate: small changes in this parameter have a drastic effect over GA performances; so it is interesting to use a controller which is able to wrap these parameters, providing a more stable operation, even by including additional control parameters.
- When the controller provides a more comprehensible abstraction of GA parameters. This is the case in our approach: it is easier for a human to think in terms of raising and lowering EEB instead of modifying multiple operators' parameters, specially when their behavior is ill-known.

The parameter τ is set to 100, and P_{min} to $\frac{1}{3k}$ (see Sect. 2.1). Each run, consisting of a specific problem instance, population size, controller and Θ (just for Compass), was replicated 30 times for significant statistical comparisons. AP and APGAIN parameters were set to published values, or tuned to obtain good performance. According to the notations used in [7,6], for AP: $\alpha = 0.8$, $\beta = 0.8$, $P_{min} = \frac{1}{2k}$. For APGAIN: $v_L = 0$, $v_U = 1$, $\delta = 0.05$, $\sigma = 700$, $\rho = \frac{\sigma}{4}$, $\xi = 10$, $\phi = 0.045$ (about 10% of re-evaluations).

4 Results and Discussion

The average number of false clauses obtained over 30 runs is shown in table 1. Comparisons were done using a student-t test with a significance level of 5%. Values are **boldfaced** when Compass outperforms UC, and *italicized* when UC is better than Compass. No font modification means that results are statistically indistinguishable. Cells are grey when Compass outperforms AP, and black when AP outperforms Compass. White cells means indistinguishability. Finally, Compass outperformed APGAIN in all cases, except in those indicated with underlined values, where results are indistinguishable. Average execution times of AP, APGAIN and Compass, relative to those of UC, are shown at the rightest

Table 1. Average false clauses and comparative execution times

popsize	control	4-blocks	aim	f1000	CBS	flat200	logistics	medium	par16	sw100-0	sw100-1	uf250	uuf250	time
	UC	12.9	3.2	52.6	5.4	38.4	16.7	7.7	124.2	23.2	9.4	8.3	11.7	1.00
	AP	7.5	2.1	37.7	3.4	19.6	8.9	3.5	71.3	16.2	3.3	5.6	8.3	0.86
	APGAIN	11.8	3.3	51.6	5.0	27.5	14.0	5.4	109.2	20.6	6.0	8.8	10.8	0.97
3	C.2	**5.8**	**2.1**	**25.5**	**2.1**	**13.2**	**8.1**	**2.0**	**41.6**	**13.7**	**1.3**	**3.3**	**5.5**	0.88
	C.25	**6.4**	**1.6**	**26.7**	**2.3**	**11.7**	**8.1**	**2.0**	**38.4**	**13.4**	**1.8**	**3.6**	**5.9**	0.89
	C.3	**6.1**	**1.6**	**26.8**	**2.8**	**15.9**	**8.1**	**3.0**	**47.2**	**15.5**	**2.2**	**4.3**	**6.4**	0.73
	UC	13.8	3.3	61.4	7.6	34.6	16.5	7.9	126.2	24.6	8.3	11.2	14.0	1.00
	AP	8.9	3.0	47.4	4.8	23.3	11.3	4.9	88.2	18.4	4.5	8.3	10.2	0.88
	APGAIN	11.2	4.9	60.1	6.6	31.1	14.0	6.4	118.7	20.1	6.0	9.8	13.6	1.09
5	C.2	**6.2**	**2.2**	**27.5**	**3.0**	**15.5**	**9.1**	**2.6**	**45.0**	**15.0**	**2.8**	**4.6**	**6.7**	0.89
	C.25	**6.3**	**1.9**	**27.2**	**2.7**	**16.2**	**8.8**	**2.6**	**43.4**	**14.8**	**2.9**	**4.4**	**7.1**	0.91
	C.3	**7.8**	**2.5**	**36.5**	**4.2**	**20.0**	**9.0**	**3.5**	**66.7**	**16.8**	**3.4**	**5.9**	**10.3**	0.80
	UC	13.8	3.3	54.2	6.3	28.8	15.5	7.0	110.0	19.4	6.1	10.1	12.2	1.00
	AP	9.9	3.9	55.2	5.0	26.3	12.9	5.7	98.6	18.1	5.9	9.3	12.2	1.00
	APGAIN	11.7	4.8	66.0	5.8	30.4	16.3	6.2	120.4	18.8	6.8	11.3	13.9	0.62
10	C.2	**8.3**	3.5	**44.9**	**3.9**	**21.9**	**11.3**	**4.5**	**72.5**	**17.9**	**4.3**	**7.5**	**10.4**	0.92
	C.25	**8.0**	2.9	**42.7**	**4.4**	**23.1**	**11.2**	**4.2**	**72.6**	**17.8**	**4.6**	**7.2**	**9.5**	0.83
	C.3	9.1	3.7	**49.3**	5.7	**24.7**	**11.5**	**5.1**	**96.8**	19.1	5.9	9.1	12.6	0.78
	UC	13.8	2.8	42.0	4.5	23.1	13.8	5.2	88.5	16.5	3.4	6.4	10.3	1.00
	AP	9.1	2.4	54.1	4.9	26.0	14.6	5.3	102.6	17.2	5.7	8.4	11.3	1.28
	APGAIN	11.3	4.3	68.0	5.5	30.0	17.8	6.2	120.1	18.7	6.3	11.1	13.7	2.38
20	C.2	**9.1**	*3.5*	*55.2*	4.8	*26.0*	13.3	5.2	90.4	*18.9*	*6.2*	*8.1*	*12.5*	0.92
	C.25	**9.2**	3.3	*53.2*	4.8	*25.2*	13.3	4.8	90.6	*17.8*	*6.2*	*8.7*	*13.2*	0.93
	C.3	**9.4**	*3.9*	*58.6*	4.9	*27.5*	13.4	6.2	*99.2*	*18.7*	*6.1*	*10.4*	*12.4*	0.88
Total clauses		47820	320	4250	449	2237	6718	953	3310	3100	3100	1065	1065	

column of the table. Total number of clauses of each problem appear in the bottom of the table. From now on we will refer as *C.2, C.25, C.3* to Compass with Θ values of 0.20π, 0.25π and 0.30π, respectively.

The mean number of generations required to reach the best values varies between 1000 and 7000, therefore, the 10000 allowed generations seem sufficient for all controllers to insure a fair comparison. Of course, the results are not competitive with specific SAT evolutionary solvers, since we do not use the best dedicated operators and neither try to optimize ours. Our purpose here is rather to highlight the differences between controllers. Better results for SAT using GAs were obtained by a hierarchical memetic algorithm [19]. However, given the early stage of this research, we preferred a simpler GA that applies one operator each time, in order to facilitate understanding. Further research will consider more complex operator architectures.

The predominance of Compass, and specially C.25 over UC, AP and APGAIN is noticeable, particularly for small populations. Something similar happens with C.2, and, to some extent, with C.3. As mentioned previously, a value $\Theta = 0.25\pi$ works well with all kind of problems.

Small populations lose diversity easily, so controlling diversity is a critical issue. APGAIN does it by penalizing common individuals. However, when all individuals are the same, this penalization is not effective. It seems that in practice, diversity is mostly induced by the first period in APGAIN (operator evaluation). AP controls diversity by defining a minimum application rate equal to $\frac{1}{2k}$. This value could be excessive if operators are mostly exploitative. A smaller value of $\frac{1}{3k}$, used in Compass, grants the controller a greater range to balance EEB.

Small populations provide better results than larger ones. This is probably due to the operators, that were inspired by local search heuristics: they are applied more repetitively over the same individual in smaller populations than in large ones, thus producing better results. Surprisingly, UC is quite competitive as population size increases. It seems that applying operators of both low d_{it} and q_{it} produce "bad" individuals that are able, however, to escape from local optima. Nevertheless, this practice is beneficial only if the population is big enough to keep their good elements at the same time.

Execution times of AP and APGAIN are shorter than those of UC for the smallest populations. Compass has stable short execution times for different population sizes. This is interesting because it means that the effort spent in performing control induces savings in total execution time.

From an implementation point of view, we found that Compass and AP were more independent from the logic of the GA than APGAIN, which introduces its diversity control mechanism in the GA fitness function. Both AP and Compass provide a separate layer of control. Parameterization of Compass is quite intuitive. We have already discussed the effect of Θ and P_{min}. The last parameter, τ, is quite stable, we have replicated experiments with several values for this parameter without detecting a considerable influence over the performances.

5 Conclusions

In this paper we have presented Compass, a GA controller that provides an abstraction of parameters and simplifies control by adjusting the level of exploration/exploitation along the search. This controller measures operators' effects over population's mean fitness, diversity and execution time. Compass is independent from the GA, in order to provide an additional control layer that could be used by other of population-based algorithms. Experiments were performed using a 6-operators GA to solve instances of the SAT problem. Results were favorably compared against a basic uniform choice and state-of-the-art controllers.

The twofold evaluation of operators (quality and diversity variation) is coherent with the guiding principles of population-based search algorithms, i.e. maximizing quality of solutions while avoiding the concentration of the population, in order to benefit from their parallel nature. By considering both measures, we observed a natural mechanism to escape from local optima.

The search direction is easily apprehensible by observing a dynamic vectorial representation, thus Compass could also be used as a tool for understanding the role of operators.

The management of nonstandard unknown operators also opens the perspective of using Compass to evaluate operators generated automatically, for example by means of Genetic Programming.

References

1. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.: Parameter Control in Evolutionary Algorithms. In: [20], pp. 19–46
2. Jong, K.D.: Parameter Setting in EAs: a 30 Year Perspective. In: [20], pp. 1–48
3. Meyer-Nieberg, S., Beyer, H.: Self-Adaptation in EAs. In: [20], pp. 47–75
4. Kee, E., Airey, S., Cyre, W.: An adaptive genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 391–397. Morgan Kaufmann, San Francisco (2001)
5. Maturana, J., Saubion, F.: Towards a generic control strategy for EAs: an adaptive fuzzy-learning approach. In: Proceedings of IEEE International Conference on Evolutionary Computation (CEC), pp. 4546–4553 (2007)
6. Wong, L., Leung, H.: A novel approach in parameter adaptation and diversity maintenance for GAs. Soft Computing 7(8), 506–515 (2003)
7. Thierens, D.: Adaptive Strategies for Operator Allocation. In: [20], pp. 77–90
8. Igel, C., Kreutz, M.: Operator adaptation in structure optimization of neural networks. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), p. 1094. Morgan Kaufmann, San Francisco (2001)
9. Lobo, F., Goldberg, D.: Decision making in a hybrid genetic algorithm. In: Proc. of IEEE Intl. Conference on Evolutionary Computation (CEC), pp. 122–125 (1997)
10. Whitacre, J., Pham, T., Sarker, R.: Use of statistical outlier detection method in adaptive evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1345–1352. ACM Press, New York (2006)
11. Eiben, A., Marchiori, E., Valkó, V.: Evolutionary algorithms with on-the-fly population size adjustment. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 41–50. Springer, Heidelberg (2004)
12. Ursem, R.: Diversity-guided evolutionary algorithms. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 462–474. Springer, Heidelberg (2002)
13. Eiben, A., Horvath, M., Kowalczyk, W., Schut, M.: Reinforcement learning for online control of evolutionary algorithms. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS (LNAI), vol. 4335, pp. 151–160. Springer, Heidelberg (2007)
14. Lis, J.: Parallel genetic algorithm with dynamic control parameter. In: Proc. of IEEE Intl. Conference on Evolutionary Computation (CEC), pp. 324–329 (1996)
15. Tsutsui, S., Fujimoto, Y., Ghosh, A.: Forking GAs: GAs with search space division schemes. Evolutionary Computation 5(1), 61–80 (1997)
16. Harik, G., Lobo, F.: A parameter-less GA. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 258–265 (1999)
17. Cook, S.A.: The complexity of theorem-proving procedures. In: STOC 1971: Proceedings of the third annual ACM symposium on Theory of computing, pp. 151–158. ACM Press, New York (1971)
18. Hoos, H., Stützle, T.: SATLIB: An Online Resource for Research on SAT, pp. 283–292. IOS Press, Amsterdam (2000), `www.satlib.org`
19. Lardeux, F., Saubion, F., Hao, J.K.: GASAT: A genetic local search algorithm for the satisfiability problem. Evolutionary Computation 14(2), 223–253 (2006)
20. Lobo, F., Lima, C., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol. 54. Springer, Heidelberg (2007)

Testing the Intermediate Disturbance Hypothesis: Effect of Asynchronous Population Incorporation on Multi-Deme Evolutionary Algorithms*

J.J. Merelo[1], A. Mora[1], P.A. Castillo[1], J.L.J. Laredo[1], L. Araujo[2], K. Sharman[3], A.I. Esparcia-Alcázar[3], E. Alfaro-Cid[3], and C. Cotta[4]

[1] Dept. ATC, Universidad de Granada, Spain
jj@merelo.net
[2] Dept. LSI, UNED, Madrid, Spain
[3] Instituto Tecnológico de Informática, Valencia, Spain
[4] Dept. LCC, Universidad de Málaga, Spain

Abstract. In P2P and volunteer computing environments, resources are not always available from the beginning to the end, getting incorporated into the experiment at any moment. Determining the best way of using these resources so that the exploration/exploitation balance is kept and used to its best effect is an important issue. The Intermediate Disturbance Hypothesis states that a moderate population disturbance (in any sense that could affect the population fitness) results in the maximum ecological diversity. In the line of this hypothesis, we will test the effect of incorporation of a second population in a two-population experiment. Experiments performed on two combinatorial optimization problems, MMDP and P-Peaks, show that the highest algorithmic effect is produced if it is done in the middle of the evolution of the first population; starting them at the same time or towards the end yields no improvement or an increase in the number of evaluations needed to reach a solution. This effect is explained in the paper, and ascribed to the *intermediate disturbance* produced by first-population immigrants in the second population.

1 Introduction

The volatility of resources is an important feature of some distributed computation environments, such as those based on P2P or voluntary computation: resources appear and disappear in a continuous and unpredictable manner. For instance, a new node might be added to an Evolving Agents (EvAg) [1] P2P distributed evolutionary computation experiment, or a new client might download the web page to start a browser-based evolutionary experiment [2,3]. Using these high-churn computing environment efficiently so that their contribution to

* Funded by the Spanish MICYT project NoHNES (Spanish Ministerio de Educación y Ciencia - TIN2007-68083) and the Junta de Andalucía P06-TIC-02025.

the common compute pool does not get lost is obviously an important issue, and rules for using the node's computing resources efficiently (or at all) have to be researched. If the evolutionary experiment is sufficiently advanced, it might be the case that computation performed in a certain way by the new node is useless, and it will be best devoted to a new experiment (or to help the experiment in a different way). The same problem arises also in other heterogeneous and asynchronous computing experiments: even if all nodes start at the same time, those with less computing power will eventually *lag behind*, falling into a *less evolved* state that might render them useless, churning out individuals whose state would have made them eliminated in other nodes whose populations are more advanced.

There are, in principle, two different ways of creating this initial population: in a completely random way, or as an (imperfect) duplicate of the existing population. If we look at the set of the two (new and old) populations as a single one, it is obvious that these two ways correspond to tipping the exploration/exploitation balance in one way or another. The introduction of a new random population and the resulting application of the crossover operator would correspond to a hyper or macromutation operator [4,5], tipping the balance towards *exploration*, while a new population generated via application of genetic operators would correspond to an *exploitation* around the point in search space that has actually been reached. In any case, it is quite clear that the result of putting individuals from an existing evolved population in common with a new random one will result in a complex interaction, with varying results depending on the problem: it might be the case that different problems or even different phases in the execution of a problem will need different strategies.

In this paper, our objective is to find out what are the effects of the incorporation of a new population, at different times, into an existing evolution problem, and to eventually propose some heuristic rules to handle it. Our expected result will be some rule of thumb about when the addition of these new populations is most profitable or, in any case, a measure of how high is its influence on the final outcome.

As far as we know, the type of asynchrony this paper deals with has not been analyzed in depth in the existing literature. Certainly, asynchronous distributed genetic algorithms have been discussed extensively, for instance Giacobini et al. studied the selection intensity in asynchronous evolutionary algorithms [6], and Alba et al. compare them with synchronous parallel distributed genetic algorithms in [7]; a similar approach applied to distributed genetic programming was presented in [8]. In general, the conclusion is that asynchrony in evolution does not affect algorithm performance; however, that conclusion applies only if all computing nodes start at the same time, which is not the case that we want to address in this work.

Cantú-Paz [9] found that the migration policy that causes the greatest reduction in total algorithmic work (expressed as total number of evaluations) is to choose as migrants the best individuals and to replace the worse individuals in the destination population, since this policy increases the selection pressure and may cause the algorithm to converge significantly faster. However, too fast a convergence can lead to the algorithm's failure, as he states referring to parallel EAs: "rapid convergence is desirable, but an excessively fast convergence may cause the EA to converge prematurely to a suboptimal solution". In fact, Alba and Troya [10] found that migration of a random string prevents the "conquest" effect in the target island for small or medium sized sub-populations. In line with this, we study here the trade-off between selection pressure and diversity when we have nodes starting at different times; and since it has been proved the best strategy, the two nodes will migrate the best individual.

The rest of the paper is organized as follows: the experimental setup is described in Section 2, with results presented in Section 3. Finally, conclusions and future work are commented in Section 4.

2 Experimental Setup

Two functions have been used for testing: the problem generator P-Peaks and the massively multimodal deceptive problem (MMDP), two of the three discrete optimization problems presented by Giacobini et al. in [11]. These problems, while being both multimodal, represent different degrees of difficulty for parallel evolutionary optimization, and will be described next.

MMDP [12] is deceptive (that is, approached via hill-climbing algorithms would lead to a suboptimal solution) composed of k subproblems of 6 bits each. Each subproblem is evaluated on the basis of its unitation as follows:

$$f(n) = \begin{cases} 1.0 & n \in \{0,6\} \\ 0.0 & n \in \{1,5\} \\ 0.360384 & n \in \{2,4\} \\ 0.640576 & n = 3 \end{cases}$$

The fitness value of a $6k$-bit string is defined as

$$f_{MMDP}(\boldsymbol{s}) = \sum_{i=0}^{k-1} f\left(\sum_{j=1}^{6} s_{6i+j}\right)$$

Note that the number of local optima is quite large (22^k), while there are only 2^k global solutions. In this paper, we consider a single instance with $k = 20$ (120 bits).

On the other hand, the P-Peaks problem is a multimodal problem generator proposed by De Jong in [13]; a P-Peaks instance is created by generating P random $N - bit$ strings where the fitness value of a string $\boldsymbol{x}$ is the number of bits that $\boldsymbol{x}$ has in common with the nearest peak divided by N.

$$f_{P-PEAKS}(\boldsymbol{x}) = \frac{1}{N} \max_{1 \le i \le p} \{N - H(\boldsymbol{x}, Peak_i)\} \tag{1}$$

where $H(\boldsymbol{x}, \boldsymbol{y})$ is the Hamming distance between binary strings $\boldsymbol{x}$ and $\boldsymbol{y}$. In the experiments made in this paper we will consider $P = 100$ and $N = 64$. Note that the optimum fitness is 1.0.

These two problems have been implemented and integrated in the public-domain `Algorithm::Evolutionary`[14] Perl library[1]. In order to simulate a parallel algorithm, the *cooperative multitasking* Perl module POE[2] has been used; each node is represented by a POE *session*. Thus, in fact, the conclusions obtained in this paper are algorithmic in nature; if runtime conclusions have to be made, this experiment should be repeated in a true parallel environment. In this

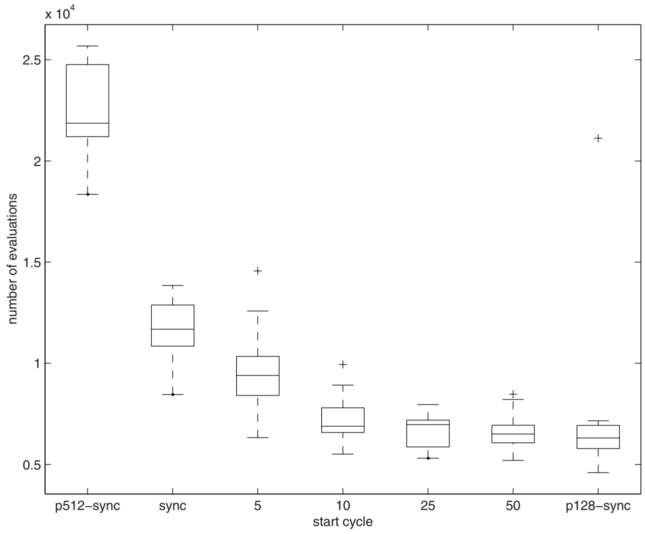


Fig. 1. Boxplot of the number of evaluations needed to find the solution in the P-Peaks problem starting at different cycles. *sync* labels cases with the two nodes starting synchronously: `p512-sync` with a population of 512, `sync` with 256, and `p128-sync` with 128 individuals; `50` represents the behavior of the experiment in a single node, since the algorithms finish before receiving any individual.

simulated parallel scenario we have implemented two nodes, each one applying a rank-based substitution steady state algorithm [15] to a single population. We do not think that using only two nodes represents a loss of generality, since migrations are always performed between only two nodes, independently of how many are running at a time. At the end of a preset number of generations (which we will call a cycle), each node sends a single individual (the best one) to the other in a theoretically synchronous manner (that is, both nodes evolve in lock-step). Algorithmic efficiency will be measured summming up the total number of evaluations performed in each node until the solution is found in one of them.

[1] Freely available under the GPL license from `http://tinyurl.com/3v4gj7`. The program, along with some configuration files and experiment results, can be downloaded from `http://tinyurl.com/4ttaow`; released versions can also be downloaded from your closest CPAN repository.

[2] Perl Object Environment; also available from CPAN.

In order to simulate the asynchronous start of the second population, several experiments were made in which the second population did not start until after a certain number of cycles. Population 1 was left running for n cycles (with g generations each), and then Population 2 started running and interchanging individuals with it. This asynchronous starting point is fixed (does not depend on the state the evolutionary algorithm is), so it could happen that Population 1 has already found the solution.

3 Experimental Results

Every configuration was run 30 times in order to obtain statistically significant results. All experiments were performed in Linux desktop and laptop machines (Ubuntu 7.04 and Fedora Core 6 and 8), with statistical analysis performed using the open source statistical package R.

For the P-Peaks experiment we have chosen the evolutionary algorithm parameters shown in Table 1 (middle column). Figure 1 shows the results of the

Table 1. Evolutionary algorithm parameters used in the P-Peaks experiments. The `Algorithm::Evolutionary` Perl library uses *priorities* for operators, that once normalized, correspond to operator rates: to 40% mutation, 60 % crossover.

Parameter	Value	
	P-Peaks	MMDP
Chromosome length	64	120
Population	256	1024
Selection rate	20%	10%
Generations to migration (cycle size)	10	
Mutation priority	2	
2-point crossover priority	3	

experiments performed with P-Peaks. For the sake of comparison, the total number of evaluations for the synchronous start experiments with population = 512 (leftmost box, labeled `p512-sync`) and population = 128 (rightmost box, labeled `p128-sync`) have also been plotted. Comparing them with the `sync` experiment (2 nodes, population = 256, synchronous start), it can be seen that lowering the population size also improves the number of evaluations (y axis). However, if we start by the `p128-sync` figure and proceed from right to left, we see that splitting the population in two (that is, going from a single population with 256 individuals – start cycle = 50[3] – to two parallel populations with 128 individuals does not yield any improvement) increases the number of evaluations needed to find the solution. Once again, moving from that experiment to its left shows what happens if, instead of letting a single population proceed, we introduce a second population by the 25th cycle (25×10 generations). What we see is a

[3] Which, in fact, would correspond to a single 256 individuals population, since by the 50th cycle, a single population has already reached the target fitness.

decrease in the quality of the algorithm, i.e., an increase of the median number of evaluations needed to reach target. A strikingly similar response is reached if the second population starts any time before that: a higher number of evaluations are going to be needed. Some other conclusions can be reached by looking at this graph from left to right, and starting by the `sync` glyph (which represents the behavior of two populations starting at the same time): whenever the second population is started *after* the first one has already run a bit of its course, the results are going to be better; however, the improvement is going to stall by the time a few cycles have already run (in this case, after the 10th cycle – 100^{th} generation, when around 50% of the runs have already finished). The Wilcoxon rank-sum test confirms that there is no difference among the four last experiments, and that the difference among the three first and the rest is significative.

Let us check these results running again the distributed evolutionary algorithm (parameters shown in Table 1, right-most column) with a more difficult problem, MMDP. The picture is quite different here, although the trend is more or less the same: there is an average trend towards decreasing the number of evaluations when the start cycle of the second population is delayed, which stops when the evolution of the first population is too advanced (in this case, after 50 cycles or 500 generations). However, the situation is not exactly the same. The main difference arises from the fact that there is a non-null set of experiments (among the

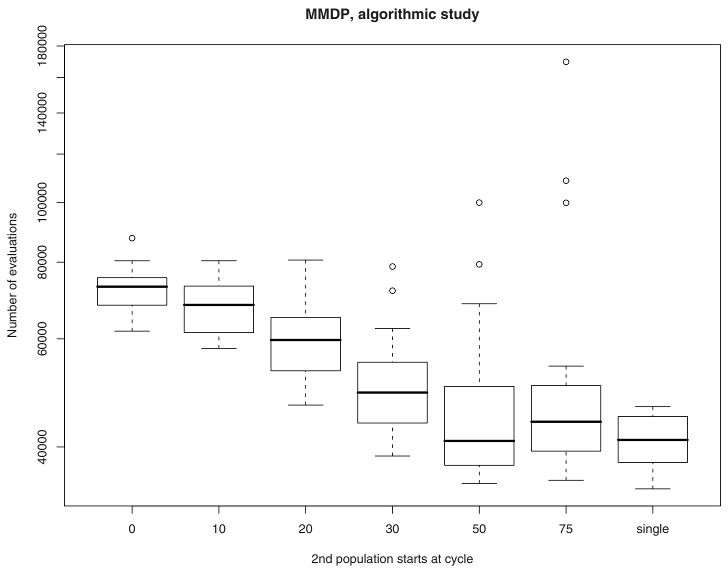


Fig. 2. Logarithmic boxplot of the number of evaluations needed to find the solution in the MMDP, after those that have not found it have been eliminated. x labels indicate the cycle when the second population has been inserted, with "single" indicating results for a single population. Once again, the Wilcoxon rank-sum test confirms the differences among the three first, and its abscense among the 4 last.

30 runs for each parameter set) that does not find the solution before the maximum number of evaluations allowed (200000). The size of this set is represented in Figure 3, which shows a rather jagged scenario, but if we look at it from right to

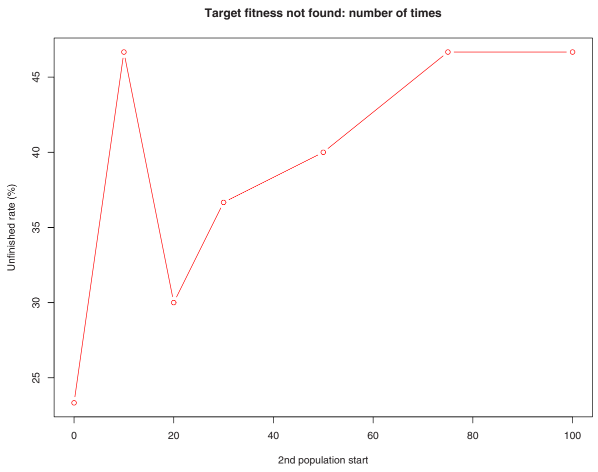


Fig. 3. Percentage of runs, for different start cycles of the second population, where the target fitness was not found in the MMDP problem

left, we see that it confirms the effect of the moment of introduction of the second population on the total quality of results: from a single population (x label $= 100$) to a late introduction of the second population ($x = 75, 50$), there is a very small improvement (from 45% to 40%). The situation gets a bit better if the second population is introduced at cycle # 20 or 30, but worsens again when it is introduced too early (cycle # 10). In this case, the best worst-case scenario is given by the synchronous population, although the best median number of evaluations is found when the second population is introduced at cycle # 30. Putting both effects together, we find that the best situation is in the *intermediate* area: lowest number of evaluations, without an excessive raise in the number of unsuccessful runs (which might be changed if the evaluation limit is set higher).

Find out the reason why this happens is a different problem, and the classical, synchronous start, two-population distributed evolutionary algorithm comes out as a worse algorithm. In order to check what is going on, we did several runs with another program where we logged the diversity after each cycle in the P-Peaks experiment. The results are plotted in Figure 4, which shows the different evolution paths of phenotypic entropy (computed using the Shannon formula) in an asynchronous (left) and synchronous (right) start experiment, in two typical cases that finished in roughly the same amount of cycles (around 30). In time, entropy tends to equalize; however, the level it reaches is that of the less diverse population (Population 1, in both cases). However, it is interesting to see that the highest effect in diversity is that of immigrants of Population 1 on Population 2; in general, the effects of the less diverse (more converged, and thus,

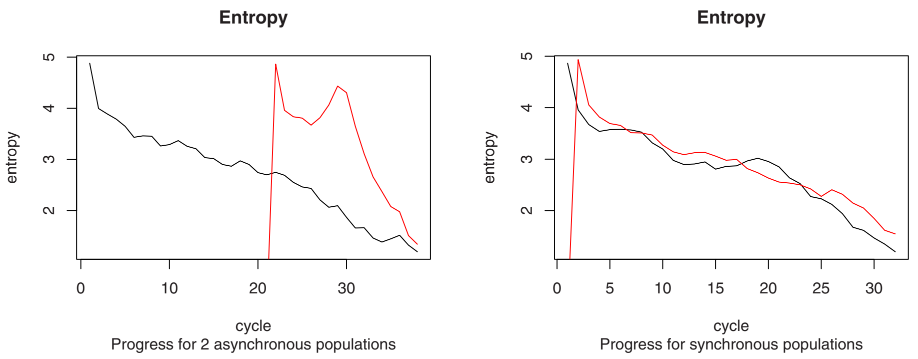


Fig. 4. Shannon Entropy ($H(P) = -\sum_{g\in P} p(f(g))log_b p(f(g))$, with g a member of the population, $f(g)$ its fitness, and $p(f(g))$ the frequency of that fitness) of both populations in a typical run of the P-Peaks problem, with asynchronous start (right) and second population starting at cycle 20 (left); Population 1 is plotted in black and Population 2 in red or light color. Please note that the total number of evaluations will be lower in the first case, since Population 2 will have performed less evaluations.

further up the evolution ladder) on the more diverse (less evolved) population. The bend found before cycle 30 in both cases indicates a quick exploitation that eventually finds the solution. This leads us to think that the reduction in the number of evaluations is mainly due to the effect of highly-fit individuals falling and eventually mating with a pool of highly-diverse ones. This effect does not take place if both populations start at the same time (diversity and fitness degrees reached are more or less the same across all the experiment), and where exploration and exploitation take place roughly synchronously (more exploitation at the beginning, more exploration at the end); or if one population is introduced too late into the simulation, where the combination of highly fit individual with low-fitness ones will amount to exploration, thus raising the total number of evaluations. However, it is the combination of highly-fit individuals with a diverse population with the right difference in fitness which produces the best algorithmic result in the shape of the best median number of evaluations.

This result is interesting, being roughly in accordance with the Intermediate Disturbance Hypothesis [16], which states that the right amount of disturbance produces the maximum diversity in ecosystems. In this case, low disturbance (migration among similar populations) and high disturbance (in-migration of an individual too highly fit) yields worse results than a better-fit individual introduced a pool of individuals that have already evolved for some time.

4 Conclusions

In this paper we have tested the influence that the introduction of a new population has on the fragile exploitation/exploration equilibrium that reigns in a single population undergoing evolution. We have used a simulated two-node parallel population, with the second population introduced at different times and tested it on two different discrete optimization problems: P-Peaks and MMDP.

The result has been rather counter-intuitive: introducing a second population a little after the first population always improves the algorithmic efficiency of the set, while doing it close to the end of evolution, as was our *a priori* hypothesis, does no good algorithmically and might even impact negatively on the overall performance. The cause of this effect has been studied and we have concluded that it might be related to the intermediate disturbance hypothesis applied to the second late-coming population (the effect of immigrants from the second population to the first being rather negligible): receiving high-fitness immigrants from the first, already evolved population will be beneficial only if the fitness difference between that immigrant and the current genetic pool is just right. If it is too high (first population highly evolved) or too low (first population started at the same time), the increase in diversity (and thus the speedup in finding the solution) in this second population will be negligible.

This yields the rule of thumb that additional populations should be started later at regular (short) intervals, instead of at the same time; and that if there is a new node arriving in a distributed computation experiment, it should be used for *outsourcing*, by doing just fitness evaluations or some other expensive task, and not for *offshoring*, by spawning a whole new population that will perform its own evolution in parallel.

In the future, we will try to confirm the results obtained in these simulations by applying it to more discrete and continuous optimization experiments, and also using more simultaneous populations, although this addition should not essentially alter the results. It would be also interesting to test them in a real parallel environment, to match not only the algorithmic gain, but also the time gain obtained by starting two populations asynchronously and in heterogeneous computers. In principle, the effect of having a second population in a slow computer would be akin to having a late-start second population, and the working hypothesis would be that the effect could be beneficial if the performance differences are not too high, but this would have to be tested. Eventually, our intention is to create a distributed computation framework that would self-adapt to late comers, asynchrony and differences in performance extracting the most from it.

References

1. Laredo, J., Castillo, P., Mora, A., Merelo, J.: Exploring population structures for locally concurrent and massively parallel evolutionary algorithms. In: [17], pp. 2610–2617
2. Merelo, J.J., García, A.M., Laredo, J.L.J., Lupión, J., Tricas, F.: Browser-based distributed evolutionary computation: performance and scaling behavior. In: GECCO 2007: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation, pp. 2851–2858. ACM Press, New York (2007)
3. Merelo, J., Castillo, P., Laredo, J., Mora, A., Prieto, A.: Asynchronous distributed genetic algorithms with Javascript and JSON. In: [17], pp. 1372–1379

4. Morrison, R., De Jong, K., Syst, M., McLean, V.: Triggered hypermutation revisited. In: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 2 (2000)
5. Jones, T.: Crossover, macromutation, and population-based search. In: Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 73–80. Morgan Kaufmann, San Francisco (1995)
6. Giacobini, M., Alba, E., Tomassini, M., et al.: Selection intensity in asynchronous cellular evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, Chicago, USA, pp. 955–966 (2003)
7. Alba, E., Troya, J.: Analyzing synchronous and asynchronous parallel distributed genetic algorithms. Future Generation Computer Systems 17(4), 451–465 (2001)
8. Fernandez, F., Galeano, G., Gomez, J.: Comparing Synchronous and Asynchronous Parallel and Distributed Genetic Programming Models. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B. (eds.) EuroGP 2002. LNCS, vol. 2278. Springer, Heidelberg (2002)
9. Cantú-Paz, E.: Migration policies, selection pressure, and parallel evolutionary algorithms. Journal of Heuristics 7(4), 311–334 (2001)
10. Alba, E., Troya, J.M.: Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. Appl. Intell. 12(3), 163–181 (2000)
11. Giacobini, M., Preuss, M., Tomassini, M.: Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In: Gottlieb, J., Raidl, G.R. (eds.) EvoCOP 2006. LNCS, vol. 3906, pp. 85–96. Springer, Heidelberg (2006)
12. Goldberg, D.E., Deb, K., Horn, J.: Massive multimodality, deception, and genetic algorithms. In: Männer, R., Manderick, B. (eds.) Parallel Problem Solving from Nature, vol. 2. Elsevier Science Publishers, B. V, Amsterdam (1992)
13. Jong, K.A.D., Potter, M.A., Spears, W.M.: Using problem generators to explore the effects of epistasis. In: Bäck, T. (ed.) Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 1997). Morgan Kaufmann, San Francisco (1997)
14. Merelo-Guervós, J.J.: Evolutionary computation in Perl. In: Perl Mongers, M. (ed.) YAPC, Europe 2002, pp. 2–22 (2002)
15. Syswerda, G.: A Study of Reproduction in Generational and Steady-State Genetic Algorithms. Foundations of Genetic Algorithms (1991)
16. Ward, J., Stanford, J.: Intermediate-Disturbance Hypothesis: An Explanation for Biotic Diversity Patterns in Lotic Ecosystems. Dynamics of Lotic Systems, Ann Arbor Science, Ann Arbor MI. 1983. 347-356 p, 2 fig, 35 ref. (1983)
17. IEEE Congress on Evolutionary Computation (CEC2008). In: WCCI 2008 Proceedings. IEEE Press, Los Alamitos (2008)

A Developmental Approach to the Uncapacitated Examination Timetabling Problem

Nelishia Pillay[1] and Wolfgang Banzhaf[2]

[1] School of Computer Science, Univesity of KwaZulu-Natal, Pietermaritzburg Campus, Pietermaritzburg, KwaZulu-Natal, South Africa
pillayn32@ukzn.ac.za
[2] Department of Computer Science, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada
banzhaf@cs.mun.ca

Abstract. The paper describes a new approach, based on cell biology, to the uncapacitated examination timetabling problem. This approach begins with a single cell which is developed into a fully grown organism through the processes of cell division, cell interaction and cell migration. The mature organism represents a solution to the particular timetabling problem. The paper discusses the performance of this method on the Carter set of benchmark problems. This data set is comprised of real-world timetabling problems. The results obtained using the developmental approach are compared to that obtained by other biologically inspired algorithms applied to the same set of benchmarks and the best results cited in the literature for the Carter data set.

Keywords: biologically inspired algorithms, uncapacitated examination timetabling problem.

1 Introduction

The examination timetabling problem involves allocating a given set of examinations to a given number of exam periods in such a manner that the hard constraints of the problem are met and the soft constraints minimized. The hard constraints and soft constraints of the problem differ from one institution to the next ([11] and [14]). The most common hard constraint is each student is not required to write more than one examination during the same period, i.e. there are no clashes. If one or more students are required to write two exams at the same time this is referred to as a clash. A timetable that meets all the hard constraints is referred to as a feasible timetable. The soft constraints of the problem tend to be contradictory and hence this value is minimized. An example of a soft constraint is that the examinations are well spread for students or that examinations for larger classes are scheduled earlier in the timetable so as to facilitate marking. The uncapacitated version of the problem does not take room capacities into consideration while the capacitated version has the added hard constraint that the number of students allocated to a particular room during a specific period must not exceed the capacity of the room.

There has been much research into finding solutions to the uncapacitated examination timetabling problem and various techniques such as tabu search, simulated annealing, constraint programming, evolutionary algorithms, ant colonization, variations of the great deluge algorithm and the variable neighborhood search algorithm have been investigated for this purpose ([14]). This paper evaluates a new biologically inspired method, namely the developmental approach (DA), as a means of finding solutions to the uncapacitated examination timetabling problem. The foundations of this methodology lie in cell biology and a solution to the problem is created by means of cell creation, cell division, cell interaction and cell migration. The DA is tested on 12 of the Carter benchmark problems and its performance on these benchmarks are compared to other biologically inspired algorithms and the best results obtained thus far for the Carter benchmarks.

The following section gives a brief account of other biologically inspired algorithms that have been applied to the uncapacitated examination timetabling problem. Section 3 presents the developmental approach and section 4 describes the methodology employed to test the performance of the DA in finding solutions to the uncapacitated examination timetabling problem. Section 5 discusses the results obtained by this method and compares these values to that produced by other biologically inspired algorithms and the best results reported for the Carter benchmarks. A summary of the findings of this study and future extensions of this work are presented in section 6.

2 Previous Work

Research into finding solutions to the uncapacitated examination timetabling problem was initiated by Carter et al. [14] who presented a heuristic-based sequential construction method with backtracking to find solutions to a number of real-world problems. This set of problems later become know as the Carter benchmark set and is generally used to compare the performance of different methodologies in solving the uncapacitated examination timetabling problem. Numerous methods including tabu search, simulated annealing, constraint programming and variable neighbourhood search have been applied to this problem. Methodologies that are currently cited in the literature as producing the best result for one or more of the Carter benchmarks include the system implemented by Caramia et al. [6], the Flex-Deluge algorithm employed by Burke et al. [4] and the hybrid system developed by Burke et al. [5].

The system implemented by Caramia et al. [6] firstly uses a greedy scheduler to allocate examinations. Examinations are scheduled in sequence according to the number of conflicts each exam is involved in. A penalty decreaser and penalty trader are used to further reduce the number of conflicts and soft constraint cost. The Flex-Deluge algorithm implemented by Burke et al. is a variation of the Great Deluge algorithm and incorporates hill-climbing. The hybrid system developed by Burke et al. [5] combines the use of variable neighbourhood search and genetic algorithms. The genetic algorithm is used to choose a set of neighbourhoods during the variable neighbourhood search.

Biological inspired methodologies that have been applied to the examination timetabling problem include memetic algorithms, evolutionary algorithms, and ant colonization. Burke et al. [3] and Ozcan et al. [12] used memetic algorithms with hill-climbing to induce timetables for the University of Nottingham and the Faculty of Engineering and Architecture at Yeditepe University respectively.

Chu et al. [7] and Shebani [16] have conducted preliminary studies on test data to investigate the effectiveness of genetic algorithms in finding solutions to the uncapacitated examination timetabling problem. Burke et al. [2] and Ross et al. [15] employ genetic algorithms to evolve solutions to the capacitated examination timetabling problem and Wong et al. [18] have used a genetic algorithm to generate a solution for Ecole de Technologie Superieure. Erben et al. [10] have implemented a steady-state grouping algorithm to evolve exam timetables for the Cater benchmark set. However, the soft constraint cost is not reported. Ulker et al. [17] evaluate the effect of employing a genetic algorithm which uses linear linkage encoding for representation purposes. This algorithm was tested on some of the Carter benchmarks with the additional objective of using the minimum number of timeslots possible.

Paquete et al. [13] employ a multi-objective evolutionary algorithm to create a timetable for the Unit of Exact and Human Sciences at the University of Algarve. Cote et al. [8] apply a hybrid multi-objective evolutionary algorithm (hMOEA) to the uncapacitated examination timetabling problem. The algorithm incorporates tabu search, variable neighborhood search and mutation operators. This algorithm has produced results comparative to the best results cited for the Carter benchmarks.

Eley [9] uses a combination of a Max-Min ant system (MMAS) and hill-climbing to find solutions to the uncapacitated examination timetabling problem. The best timetable constructed by *m* ants during *n* cycles is further improved using hill-climbing. This system was used to generate solutions to the Carter benchmarks.

Azimi [1] compares the performance of simulated annealing, tabu search, genetic algorithms and ant colonization on a number of generated data sets for the examination timetabling problem. Ant colonization and tabu search were found to perform better than the other methodologies.

The studies relevant to that presented in this paper are those conducted by Cote et al. [8] and Eley [9] as their methodologies have been tested on the same version of the Carter benchmarks and use the same objective function as that used in the study presented in this paper. A number of the studies described in this section have either solved this problem for specific schools and the data sets are not available. Section 5 compares the performance of the DA to the hMOEA system and the MMAS system.

3 The Developmental Approach (DA)

The developmental approach creates a population of organisms, with each organism being developed by mimicking processes from cell biology. Each organism represents an examination timetable with each cell corresponding to a timetable period. In this study a population size of a hundred is used. The organism with the lowest hard constraint (although we do aim for a hard constraint cost of zero, due to the randomness associated with the method we may not always get a feasible timetable) and soft constraint cost is reported as the solution. The algorithm employed to create an organism is depicted in Fig. 1.

```
Procedure Create_Organism()
Begin
    Sort the examinations in ascending order according to saturation degree
    Create a single cell and add the exam with lowest saturation degree to it
    While there are still examinations to be allocated
    Begin
      Sort the remaining examinations in ascending order according to
      saturation degree
      If there are two or more cells perform cell migration
      Determine the cost of adding the exam with the lowest saturation degree
      to each of the cells created thus far
      If there is one or more clash-free cell/s available
        Add the exam to the cell with the minimum soft constraint cost
      Else if the maximum number of cells permitted is not reached
        Perform cell division
      Else
        Randomly allocate the exam to an existing cell
      Perform cell interaction
    EndWhile
    Perform cell migration
End
```

Fig. 1. Algorithm to create an organism

Examinations are firstly sorted according to their saturation degree, i.e. the number of clash-free cells available for the exam. The overall process begins with the creation of a single cell. The examination with the lowest saturation degree is allocated to this cell. The position of the cell in the timetable is randomly chosen. If more than one clash-free cell is available when allocating an exam, the exam is added to the cell with the lowest soft constraint cost.

Cell division occurs if there are no available clash-free cells for a particular examination. In this case the parent cell divides into two daughter cells with one cell containing the exam causing the clash and the other cell contains the rest of the examinations. If the maximum number of permitted cells has already been reached cell division cannot occur and the examination is randomly allocated to an existing cell.

Cell migration involves the movement of a cell from one region of an organism to another. In the context of examination timetabling, cell migration results in the position of the cell in the timetable being changed. During cell creation and division the position of each cell in the timetable is randomly chosen. Two types of cell migration have been studied, namely, random migration and stimulus-driven migration. In random migration the position of a cell is randomly changed to a position not yet allocated or swapped with the position of an existing cell. In stimulus-driven migration the swap or change in position only takes place if it results in an improvement in the quality of the organism, i.e. a reduction in the soft constraint cost of the timetable that the organism represents. Preliminary studies found stimulus-driven migration to be more effective than random migration and hence stimulus-driven migration is used in this study. Cell migration takes place during the development process as soon as the organism contains at least two cells. Once a complete organism has been created, it goes through a process of maturation which is basically a single iteration of cell migration. Fig.2. illustrates this process.

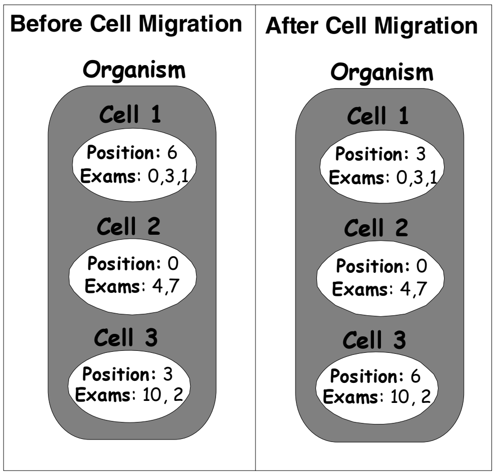


Fig. 2. Cell migration

The positions of *Cell1* and *Cell3* have been swapped as this leads to a decrease in the soft constraint cost with no increase in the hard constraint cost. Alternatively, the position of a cell could be changed to a position not yet used, e.g. 1, if this reduces the soft constraint cost of the organism.

Cell interaction involves an exchange between cells as a result of a chemical stimulus. In the context of examination timetabling the stimulus is a reduction in the soft constraint cost. Cell interaction occurs on each iteration of the development process and involves looking a the contents of each cell and determining if a change in the cell of an exam will result in a decrease in the hard constraint and soft constraint cost with the hard constraint cost having priority over the soft constraint cost. The overall process is depicted in Fig. 3.

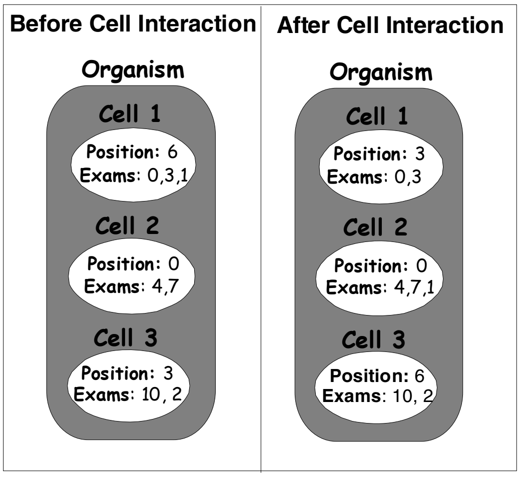


Fig. 3. Cell Interaction

Examination *1* has been moved from C*ell1* to *Cell2* as this change results in an improvement in the soft constraint cost with an improvement or no change to the hard constraint cost.

4 Experimental Setup

The developmental method was tested on the set of Carter benchmarks listed in Table 1 below.

Table 1. Carter Benchmarks

Data	Institution	Periods	No. of Exams	No. of Students	Density of Conflict Matrix
car-f-92 I	Carleton University, Ottawa	32	543	18419	0.14
car-s-91 I	Carleton University, Ottawa	35	682	16925	0.13
ear-f-83 I	Earl Haig Collegiate Institute, Toronto	24	190	1125	0.27
hec-s-92 I	Ecole des Hautes Etudes Commerciales, Montreal	18	81	2823	0.42
kfu-s-93	King Fahd University of Petroleum and Minerals, Dharan	20	461	5349	0.06
lse-f-91	London School of Economics	18	381	2726	0.06
rye-s-93	Ryerson University, Toronto	23	486	11483	0.08
sta-f-83 I	St Andrew's Junior High School, Toronto	13	139	611	0.14
tre-s-92	Trent University, Peterborough, Ontario	23	261	4360	0.18
uta-s-92 I	Faculty of Arts and Sciences, University of Toronto	35	622	21266	0.13
ute-s-92	Faculty of Engineering, University of Toronto	10	184	2749	0.08
yor-f-83 I	York Mills Collegiate Institute, Toronto	21	181	941	0.29

Note that for some of the data sets more than one version exists, thus the version is also indicated, e.g. *car-s-91I*. The density of the conflict matrix is an estimate of the difficulty of the problem and is the ratio of the number of examinations involved in clashes and the total number of examinations.

The hard constraint for the set of benchmarks is that there are no clashes, i.e. each student must not be scheduled to sit more than one exam in a given timeslot. Thus, the hard constraint cost for this problem is the number of clashes. A feasible timetable is one in which the hard constraint cost is zero, i.e. there are no clashes.

The soft constraint for each of the data sets is that the examinations must be widely spread for each student. The soft constraint cost is a measure of the quality of the timetable and we aim to minimize this value. The soft constraint cost is calculated using equation 1 [14]:

$$\frac{\sum w(|e_i - e_j|)N_{ij}}{S} \tag{1}$$

where:

1) $|e_i - e_j|$ is the distance between the periods of each pair of examinations (e_i,e_j) with common students.
2) N_{ij} is the number of students common to both examinations.
3) S is the total number of students
4) w(1) = 16, w(2) = 8, w(3) = 4, w(4) = 2 and w(5) = 1, i.e. the smaller the distance between periods the higher the weight allocated.

The system was implemented in Java and simulations were run on a Windows XP machine with a 3000 Mhz Intel 4 HT processor.

5 Results and Discussion

The DA was able to induce a feasible timetable for all 12 of the data sets. Table 2 lists the best result obtained by the developmental approach over ten runs for each data set. The best timetable generated for each data set can be found at http://saturn.cs.unp.ac.za/~nelishiap/et/da-ue.htm.

The runtime of the system varied from less than two minutes for the smaller data sets such as *hec-s*-92 to about 22 hours for the larger data sets such as *car-f-92*and *car-s-91.* Future extensions of the project will investigate ways to reduce the runtime for larger data sets. The table also lists the best results obtained by other biological inspired algorithms applied to the same set of benchmark problems, namely, the hybrid multi-objective evolutionary algorithm implemented by Cote et al.[8] and the Max-Min ant system (MMAS) used by Eley[9]. Both these systems are described in section 2. As all three methods have found feasible timetables, note that the best result is defined in terms of the quality of the timetable, i.e. the soft constraint cost. This cost is calculated using equation 1 defined in section 4. As the three different methods were run on machines with different technical specifications a comparison of the runtime is not presented. Furthermore, the methodologies that the DA is being compared to employ very different search mechanisms from that used by the system and a direct comparison of the parameters, such as the number of runs, used is therefore not feasible.

The results obtained by the developmental approach are comparative to that obtained by the other biologically inspired methods. The last column of Table 2 lists the difference of the best soft constraint cost obtained by the DA and the best soft constraint cost obtained over all three biologically inspired algorithms. The developmental approach performed better than the other biologically inspired algorithms on six of the data sets. Furthermore, for the remaining data sets the results obtained by the DA are within range of the best result.

Table 2. Performance of the DA and other biological inspired algorithms on the Carter benchmarks

Data Set	DA	hMOEA	MMAS	Difference
car-f-92 I	**4.1**	4.2	4.8	-
car-s-91 I	**5.0**	5.4	5.7	-
ear-f-83 I	35.09	**34.2**	36.8	0.89
hec-s-92 I	11.08	**10.4**	11.3	0.68
kfu-s-93	**14.1**	14.3	15.0	-
lse-f-91	**10.59**	11.3	12.1	-
rye-s-93	9.17	**8.8**	10.2	0.37
sta-f-83 I	157.28	**157.0**	157.2	0.28
tre-s-92	**8.33**	8.6	8.8	-
uta-s-92 I	**3.31**	3.5	3.8	-
ute-s-92	26.5	**25.3**	27.7	1.2
yor-f-83 I	39.4	**36.4**	39.6	3

The methods that have produced the best quality timetable for one or more of the same version of the Carter benchmarks have been discussed in Section 2. Table 3 compares the best results obtained by the developmental method with the best result cited for each of the data sets. The difference in these values is listed in the last column of Table 3. It is evident from Table 3 that the results obtained by the developmental approach are very close to the best results cited for each of the benchmarks.

Table 3. Performance of the DA and the best results cited for the Carter benchmarks

Data Set	DA	Caramia et a. [6]	Burke et al. [4]	Burke et al. [5]	Difference
car-f-92 I	4.1	6.0	4.42	**3.9**	0.2
car-s-91 I	5.0	6.6	**3.74**	4.6	1.26
ear-f-83 I	35.09	**29.3**	32.76	32.8	5.79
hec-s-92 I	11.08	**9.2**	10.15	10.0	1.88
kfu-s-93	14.15	13.8	**12.96**	13.0	1.19
lse-f-91	10.59	**9.6**	9.83	10.0	0.99
rye-s-93	9.17	**6.8**	-	-	2.37
sta-f-83 I	157.28	158.2	157.03	**156.9**	0.38
tre-s-92	8.33	9.4	**7.75**	7.9	0.58
uta-s-92 I	3.31	3.5	**3.06**	3.2	0.25
ute-s-92	26.5	**24.4**	24.82	24.8	2.1
yor-f-83 I	39.4	36.2	**34.84**	34.9	4.56

6 Conclusion and Future Work

The main aim of the study presented in this paper is to test a new developmental approach, based on cell biology, to the uncapacitated examination timetabling problem. The developmental approach has performed well on the 12 Carter benchmarks. The results produced by the DA are comparative to those produced by other biologically inspired algorithms applied to the same set of benchmark problems and has performed better than these algorithms on six of the problems. Furthermore, the results are within range of the best results cited for the benchmark set.

This study has clearly established the potential of the developmental approach. Future work will focus on further refining this methodology so as to improve both the quality of solutions produced and the runtime of the system. The processes of cell migration and cell interaction will be studied in detail to establish the effect that these processes have on the overall approach. One of the reasons for the long runtimes for the larger data sets is that all of the cells are involved in cell migration and cell interaction and both these processes are implemented on each iteration of the development of an organism. Investigations into the impact of this and effective frequencies for the application of cell migration and interaction will be conducted. In the current version of the system, if a clash-free cell cannot be found for a particular examination and the maximum number of cells has been reached the exam is added to a randomly chosen cell which will result in a clash. Future extensions of this study will examine a form of cell interaction to remove such a clash. Furthermore, a more constrained set of problems have been made available by the organizers of the 2nd International Timetabling Competition (http://www.cs.qub.ac.uk/itc2007) and a variation of the DA has been applied to these problems and is currently being refined.

Acknowledgments. The authors would like to thank the reviewers for their helpful comments and suggestions.

References

1. Azimi, Z.N.: Comparison of Metaheuristic Algorithms for Examination Timetabling Problem. Journal of Mathematics and Computing 16, 337–354 (2004)
2. Burke, E.K., Elliman, D., Weare, R.: A Genetic Algorithm Based University Timetabling System. In: Proceedings of the 2nd East-West International Conference on Computers in Education, vol. 1, pp. 35–40 (1994)
3. Burke, E.K., Newall, J.P., Weare, R.F.: A Memetic Algorithm for University Timetabling. In: Burke, E.K., Ross, P. (eds.) PATAT 1995. LNCS, vol. 1153, pp. 241–250. Springer, Heidelberg (1996)
4. Burke, E.K., Bykov, Y.: Solving Exam Timetabling Problems with the Flex-Deluge Algorithm. In: Burke, E.K., Rudova, H. (eds.) Proceedings of the International Conference on the Theory and and Practice of Automated Timetabling (PATAT 2006), pp. 370–372 (2006)
5. Burke, E.K., Eckersley, A., McCollum, B., Petrovic, B., Qu, R.: Hybrid Variable Neighborhood Approaches to University Exam Timetabling. Technical Report NOTTCS-TR-2006-2. School of Computer Science and Information Technology, Nottingham, UK (2006)
6. Caramia, M., Dell Olmo, P., Italiano, G.F.: Novel Local-Search-Based Approaches to University Examination Timetabling. INFORMS Journal of Computing 20(1), 86–99 (2008)
7. Chu, S.C., Fang, S.L.: Genetic Algorithms vs. Tabu Search in Timetable Scheduling. In: Proceedings of the 3rd International Conference on Knowledge-Based Intelligence Information Engineering Systems, pp. 492–495. IEEE Press, Los Alamitos (1999)

8. Cote, P., Wong, T., Sabourin, L.: Application of a Hybrid Multi-Objective Evolutionary Algorithm to the Uncapacitated Exam Proximity Problem. In: Burke, E.K., Trick, M. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 108–121. Springer, Heidelberg (2005)
9. Eley, M.: Ant Algorithms for the Exam Timetabling Problem. In: Burke, E.K., Rudová, H. (eds.) PATAT 2007. LNCS, vol. 3867, pp. 364–382. Springer, Heidelberg (2007)
10. Erben, W., Song, P.Y.: A Hybrid Grouping Genetic Algorithm for Examination Timetabling. In: Burke, E.K., Trick, M. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 487–490. Springer, Heidelberg (2005)
11. McCollum, B.: A Perspective on Bridging the Gap between Theory and Practice in University Timetabling. In: Burke, E.K., Rudová, H. (eds.) PATAT 2007. LNCS, vol. 3867, pp. 3–23. Springer, Heidelberg (2007)
12. Ozcan, E., Ersoy, E.: Final Exam Scheduler – FES. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1356–1363. IEEE Press, Los Alamitos (2005)
13. Paquete, L.F., Fonseca, C.M.: A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms. In: Proceedings of the 4th Metaheuristics Conference (MIC 2001), pp. 149–154 (2001)
14. Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G., Lee, S.Y.: A Survey of Search Methodologies and Automated Approaches for Examination Timetabling. Computer Science Technical Report No. NOTTCS-TR-2006-4, UK (2006)
15. Ross, P., Hart, E., Corne, D.: Some Observations about GA-Based Exam Timetabling. In: Burke, E.K., Carter, M. (eds.) PATAT 1997. LNCS, vol. 1408, pp. 115–130. Springer, Heidelberg (1998)
16. Shebani, K.: An Evolutionary Approach for the Examination Timetabling Problems. In: Burke, E.K., De Causmaecker, P. (eds.) Proceedings of the 4th International Conference on the Theory and Practice for Automated Timetabling, pp. 387–396 (2002)
17. Ulker, O., Ozcan, E., Korkmaz, E.: Linear Linkage Encoding in Grouping Problems. In: Burke, E.K., Rudová, H. (eds.) PATAT 2007. LNCS, vol. 3867, pp. 347–363. Springer, Heidelberg (2007)
18. Wong, T., Cote, P., Gely, P.: Final Exam Timetabling: A Practical Approach. In: IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2002), vol. 2, pp. 726–731. IEEE Press, Los Alamitos (2002)

QFCS: A Fuzzy LCS in Continuous Multi-step Environments with Continuous Vector Actions

José Ramírez-Ruiz, Manuel Valenzuela-Rendón, and Hugo Terashima-Marín

Center for Intelligent Systems
Tecnológico de Monterrey, Campus Monterrey
64849 Monterrey, N.L., Mexico
{a00792432,valenzuela,terashima}@itesm.mx

Abstract. This paper introduces the QFCS, a new approach to fuzzy learning classifier systems. QFCS can solve the multistep reinforcement learning problem in continuous environments and with a set of continuous vector actions. Rules in the QFCS are small fuzzy systems. QFCS uses a Q-learning algorithm to learn the mapping between inputs and outputs. This paper presents results that show that QFCS can evolve rules to represent only those parts of the input and action space where the expected values are important for making decisions. Results for the QFCS are compared with those obtained by Q-learning with a high discretization to show that the new approach converges in a way similar to how Q-learning does for one-dimension problems with an optimal solution, and for two dimensions QFCS learns suboptimal solutions while it is difficult for Q-learning to converge due to that high discretization.

Keywords: Learning Classifier Systems, Fuzzy Classifier Systems, Fuzzy Logic, Genetic Algorithm, Induction Theory.

1 Introduction

Many works have been done around Learning Classifier Systems (LCSs) since their introduction by Holland [1]. These systems learn by reinforcement to solve problems using a set of rules that compete to determine the system's behavior. Periodically, bad rules are replaced for possible better ones by a steady-state genetic algorithm (GA). In [2] Wilson introduced XCS, a LCS that uses a modified Q-learning [3,4] algorithm to learn a mapping between inputs and outputs, and the GA evolves rules based on their accuracy to represent that mapping. Wilson showed that XCS learns rules that represent maximal generalizations. XCS was created to act in discrete problems, but many real world problems are not discrete, therefore it is necessary to look for new algorithms that can tackle continuous problems.

Modifications to the XCS to handle continuous inputs have been previously proposed [5,6,7]. These were tested in the real n-multiplexor problem. This is a problem with continuous inputs, but with a set of discrete actions; also the real n-multiplexer is a one-step action problem.

The XCSF is an XCS modified to learn continuous functions [8]. XCSF uses the same machinery of the XCS, but with a change in rule representation. Rules are activated over hyper-rectangular regions of the input space, and the values of the expected payoff are calculated by adjusting a hyper-linear function by a modified delta rule. The function is obtained from the learned mapping. The XCSF has only one action that is not used to generate the output of the system, therefore, it does not make rules compete. Lanzi et al. applied the XCSF to a continuous navigation task in one dimension (the *continuous linear corridor problem*) and two dimensions (the *2D continuous gridWorld problem*) [9], In these learning tasks, XCSF perceives a continuous input and chooses an action from a set of discrete actions.

Wilson [10] proposes three different architectures that use combinations of two XCSFs to deal with continuous inputs and outputs in a simple problem called the *frog problem,* in which the system is reset after each action, and thus, it is a one-step problem. This problem was also tackled modifying the XCSF [11] to use two GAs.

LCSs that apply fuzzy logic (FL) have been studied [12] because of its capability to represent continuous variables. The first approaches [13,14,15] learned functions by reinforcement. They use cooperation instead of competition among rules to determine outputs. Bonarini presented a fuzzy classifier system that combined competition and cooperation among rules to learn an action function in multistep environments with continuous reward [16]. This methodology was applied to a CAT robot so that it would learn to move through a corridor using a continuous reward function and with different learning schemes, such as the bucket brigade algorithm, temporal differences and Q-learning [17,18]. These works produced continuous actions from a continuous perception but with an exponential number of fuzzy states to represent the perception state of the robot.

Reinforcement learning with continuous inputs and outputs is still an unsolved problem for learning classifier systems. This paper proposes a new approach to FCSs that uses Q-learning and FL in the rule representation to solve the multistep problem with continuous inputs and outputs. The proposed approach was tested on continuous navigation tasks in one and two dimensions with vectors as actions. It was also applied to the frog problem defined in [10,11] and compared with Q-learning with a high discretization.

2 QFCS

In QFCS, each classifier contains a small fuzzy system (SFS), a matrix that contains the expected prediction, and a square subregion of activation over the input space. The classifiers can only act over their subregion of activation. In that subregion of activation, each fuzzy system proposes a continuous vector field as an action by defuzzification. In that way, when an input vector enters the QFCS, classifiers compete to place their actions according to their expected predictions. A Q-learning algorithm is used to learn the task from the environment, i.e., the learning of the continuous vector action function. This algorithm is used to change the values of the matrices of the expected predictions for each classifier in the QFCS. A GA is applied to evolve rules based on their average expected predictions. This GA only evolves the action parts of the fuzzy systems, therefore there not exists generalization mechanism.

The components of the QFCS are a set of N classifiers, a performance component, a learning component, and a discovery component:

Classifiers. Each classifier Cl_i contains a SFS over the input $(x_1, \ldots, x_n)$ and output $(y_1, \ldots, y_m)$ spaces, and a square region R^i over that input space defined as

$$\left\{(x_1, \ldots, x_n) \mid \min^i_{x_1} \le x_1 \le \max^i_{x_1}, \ldots, \min^i_{x_n} \le x_n \le \max^i_{x_n}\right\} \qquad (1)$$

where $\min^i_{x_k}$ and $\max^i_{x_k}$ are constants, and a set of elements $p^i_{b_1,\ldots,b_n}$ with the subscrips $b_k \in \{1, \ldots, b\}$, $k = 1, \ldots, n$ and b a constant that constitutes an n-dimentional matrix. The classifiers can only act over their regions R^i. To form these regions, the whole input space is divided into d^n (d to the power of n) uniform square regions $R_{d_1,\ldots,d_n}$ with the subscrips $d_l \in \{1, \ldots, d\}$, $l = 1, \ldots, n$ and d a constant that determines the number of divitions per dimention. Then, classifiers can only take one out of them. Each $R_{d_1,\ldots,d_n}$ is again divided into b^n (b to the power of n) uniform square regions $\Delta_{b_1,\ldots,b_n}$ that are associated one by one with $p^i_{b_1,\ldots,b_n}$.

Performance Component. This component determines the procedure the QFCS follows to select actions as responses to the inputs. First, it receives a real vector $\boldsymbol{x}_0$ as input; then, all classifiers that contain the input in their own regions are activated to form a match set $[M]$. At that time, the input is fuzzified and is introduced into each SFS of the classifiers in $[M]$. The fuzzy inference machine of each SFS works to produce an output fuzzy set that is defuzzified into an output vector $\boldsymbol{y}_i = (y_1, \ldots, y_m)$ proposed by each classifier Cl_i in $[M]$. Then, one of those $\boldsymbol{y}_i$s is selected as the output of the QFCS $\boldsymbol{y}_0$ in the following manner:

$$\boldsymbol{y}_0 = \left\{ \max_{\boldsymbol{y}_i \in [M]} \left\{p^i_{b_1,\ldots,b_n}\right\} \mid \boldsymbol{x}_0 \in \Delta^i_{b_1,\ldots,b_n} \right\} \qquad (2)$$

Learning Component. This component makes the QFCS learn the task using Q-learning. QFCS receives, from the environment and for each input, a reward $R(\boldsymbol{x})$ that defines the task to achieve. At each time, QFCS decides what to do exploiting the knowledge it has acquired or exploring new possible actions. The exploitation is done with probability $P_{\text{exploitation}}$, while exploration is done with probability $P_{\text{exploratation}} = 1 - P_{\text{exploitation}}$. *Exploitation* is achieved by selecting $\boldsymbol{y}_0$ as in the performance component, and *exploration* by selecting one of the possible $\boldsymbol{y}_i \in [M]$ at random. Each time, $p^i_{b_1,\ldots,b_n}$ is adjusted as follows:

$$p^i_{b_1,\ldots,b_n,t-1} \leftarrow p^i_{b_1,\ldots,b_n,t-1} + \beta\left[\left(R\left(\boldsymbol{x}_{t-1}\right) + \gamma p^i_{b_1,\ldots,b_n,t}\right) - p^i_{b_1,\ldots,b_n,t-1}\right] \quad (3)$$

where $\beta, \gamma \in [0,1]$ and

$$p^i_{b_1,\ldots,b_n,t-1} = \left\{p^i_{b_1,\ldots,b_n} \mid \boldsymbol{x}_{t-1} \in \Delta^i_{b_1,\ldots,b_n} \wedge Cl_i = Cl_{t-1}\right\} \tag{4}$$

$$p^i_{b_1,\ldots,b_n,t} = \left\{p^i_{b_1,\ldots,b_n} \mid \boldsymbol{x}_{t} \in \Delta^i_{b_1,\ldots,b_n} \wedge Cl_i = Cl_{t}\right\} \tag{5}$$

with $\boldsymbol{x}_{t-1}$ and Cl_{t-1} that are the input and the classifier that was selected by the performance component at time $(t-1)$, and $\boldsymbol{x}_t$ and Cl_t that are the input and the classifier that would be selected by exploitation (the one that has the maximun expected prediction for the corresponding input) at time t. β is variable throughout time depending on how old δ_i, the classifier Cl_i, is. In other words:

$$\beta = \begin{cases} \left[\frac{\beta_0 - 1}{\delta_0}\right]\delta_i + 1, \text{ if } \delta_i < \delta_0; \\ \beta_0, \qquad \text{otherwise;} \end{cases} \tag{6}$$

where δ_0 is a constant. The age δ_i of the classifiers starts in 0 and is incremented in one unit every time step.

Discovery Component. QFCS uses a GA to create new rules. The GA is applied over $[M]$. It only evolves the action parts of the fuzzy rules of the SFSs. First, the $\overline{p}_i$ value of each $Cl_i \in [M]$ is computed by averaging their expected values $p^i_{b_1,\ldots,b_n}$. Then the GA takes two classifiers from $[M]$ selecting them with probability proportional to their $\overline{p}_i$. These classifiers are copied, crossed-over with probability χ, and mutated with probability μ. Crossing applies one-point crossover. Then, one of the two classifiers is inserted in $[M]$ replacing another one in $[M]$ that is selected proportional to $c/\overline{p}_i$. The GA is applied over $[M]$ on time intervals that are determined by the classifiers in $[M]$. For this, each classifier stores the last time e_i in which it was involved in a GA, so when $[M]$ happens, the average time $\overline{e}$ of its classifiers, for the last application of the GA, is calculated by:

$$\overline{e} = \frac{\sum\limits_{Cl_i \in [M]} (t - e_i)}{|[M]|}, \tag{7}$$

where $|[M]|$ represents the number of classifiers in $[M]$. If $\overline{e} \geq \theta_{GA}$ the GA is applied.

3 Structure, Parameters, and Experiments of QFCS

In all the experiments the input space is divided in regions $R_{d_1,\ldots,d_n}$ with $d = 4$ and with regions $R_{b_1,\ldots,b_n}$ with $b = 5$. The SFSs of classifiers are defined over their regions $R_{d_1,\ldots,d_n}$ and the output space. Therefore, two fuzzy sets $A^i_{x_q,1}$ and $A^i_{x_q,2}$ are defined with triangular membership functions per input variable x_q as follows:

$$\mu_{A^i_{x_q,p}}(x_q) = \begin{cases} \left[\frac{2}{w}\right]x + \left[1 - \frac{2c^i_p}{w}\right], \text{ if } \left[c^i_p - \frac{w}{2}\right] \leq x < c^i_p; \\ -\left[\frac{2}{w}\right]x + \left[1 + \frac{2c^i_p}{w}\right], \text{ if } c^i_p \leq x < \left[c^i_p + \frac{w}{2}\right]; \\ 0, \qquad \text{otherwise;} \end{cases} \tag{8}$$

where $q \in \{1, \ldots, n\}$, $p \in \{1, 2\}$, $w = (\max^i_{x_q} - \min^i_{x_q})$, $c^i_1 = \min^i_{x_q}$ and $c^i_1 = \max^i_{x_q}$. In the output variables y_q, the fuzzy sets $B^i_{y_q,h}$ are singletons defined by:

$$\mu_{B^i_{y_q,h}}(y_q) = \begin{cases} 1, \text{ if } y = y_{0_h}; \\ 0, \text{ otherwise;} \end{cases} \tag{9}$$

where $h \in \{1, \ldots, 5\}$ and

$$y_{0h} = \min_{y_q} + (h-1)\left[\frac{\max_{y_q} - \min_{y_q}}{4}\right], \tag{10}$$

where $\min_{y_q}$ and $\max_{y_q}$ are the lower and the upper limits of y_q. By doing this, the fuzzy rules have the next form:

$$\text{IF } \left[X^i_1 = A^i_1 \wedge \ldots \wedge X^i_n = A^i_n\right] \text{ THEN } \left[Y^i_1 = B^i_1, \ldots, Y^i_m = B^i_m\right] \tag{11}$$

where $A^i_{x_q} \in \{A^i_{x_q,p}\}$ and $B^i_{y_q}$ is a fuzzy disjunction (with maximum) of a subset S of the fuzzy sets in $T = \{B^i_{y_q,h}\}$. In the starting settings each possible fuzzy set in T has a probability of 0.05 of being in S. The SFSs of each classifier use all the possible combinations of their input fuzzy sets in the condition parts of their fuzzy rules, therefore, there are $2n$ possible fuzzy rules in each classifier Cl_i. Each SFS uses the minimum inference engine [12] with generalized modus ponens inference and with Mamdani's minimum implication. The input is fuzzified into an n-dimension singleton. The fuzzy output is defuzzified by the center of gravity. In the learning component $\beta_0 = 0.2$, $\gamma = 0.1$, $P_{\text{exploitation}} = 0.7$ and $P_{\text{exploration}} = 0.3$. In the discovery component $c = 0.5$, $\chi = 0.8$ and $\mu = 0.04$. The rest of the parameters θ_{GA}, N, n, m, and δ_i depend on the problem.

In our experiments, similarly to Lanzi et al. [9], we compared the results of QFCS with those obtained using Q-learning with a high discretization. During a learning run, QFCS is allowed a series of trials in the problem. Each trial consists of 200 steps or less if the QFCS reaches the goal before that number of steps. The number of steps given to reach the goal against trials during a run is reported. These curves are an average over 20 runs of the QFCS with the same problem. First, QFCS was proved on the frog problem introduced in [10]. Then, the QFCS was evaluated on the n-environment problem, which we propose as a more general learning task that requires continuous actions.

4 The Frog Problem

The *frog problem* [10,11] consists on a frog that lives in a continuous one-dimensional space $x \in [0, 2]$, and that can jump a distance a, i.e., its action, where $a \in [0, 1]$. A fly is placed on $x_{\text{fly}} = 1$ and the frog is placed in a position x where $x \in [0, 1]$. The goal for the frog is to jump once and catch the fly. That is considered a trial. The frog receives a reward given by:

$$R(x, a) = \begin{cases} x + a, & \text{if } x + a \leq 1; \\ 2 - (x + a), & \text{otherwise.} \end{cases} \tag{12}$$

To solve this problem, QFCS identifies the variables x_1 and y_1, with x and a respectively. We set the QFCS with a classifiers population of $N = 200$, $\theta_{GA} = 1000$, $n = m = 1$, $\delta_0 = 500$ and performed $500,000$ trials per run. Figure 1 shows the results obtained by QFCS and Q-learning. Q-learning was discretized over 100 elements in x and a variables as in [9]. The results show that QFCS learns a good approximation to the optimal continuous action with a smaller error than Q-learning. Q-learning does not learn the optimal solution due to its discretization.

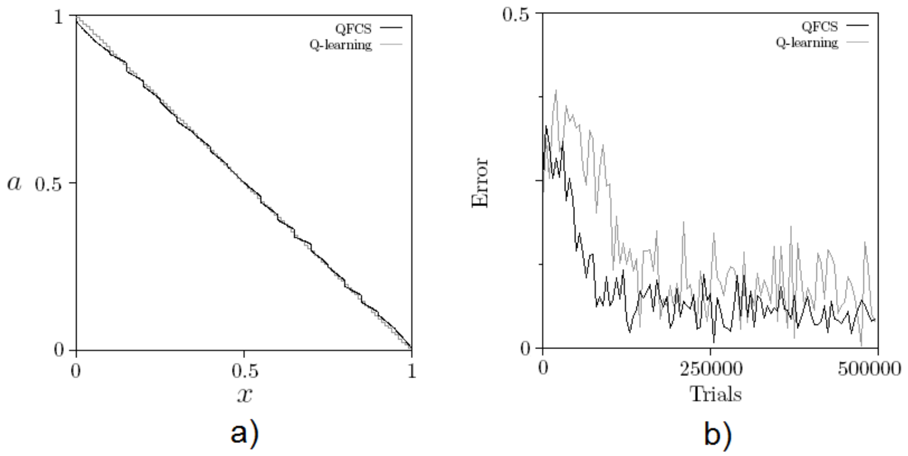


Fig. 1. (a) Average over 20 runs for the action learned. (b) Average over 20 runs for the obtained error during learning.

5 The *n*-Environment Problem

The n-environment is an n-dimensional continuous space that is determined by a square region:

$$\boldsymbol{x} = \left\{(x_1, \ldots, x_n) \mid x_1^{\min} \leq x_1 \leq x_1^{\max}, \ldots, x_n^{\min} \leq x_1 \leq x_n^{\max}\right\} \tag{13}$$

where $x_i^{\min}$ and $x_i^{\max}$ represent the lower and the upper limits of the x_i variable. This environment has a set of n-dimensional continuous action vectors defined as:

$$\boldsymbol{a} = \left\{(a_1, \ldots, a_n) \mid a_1^{\min} \leq a_1 \leq a_1^{\max}, \ldots, a_n^{\min} \leq a_1 \leq a_n^{\max}\right\}, \tag{14}$$

where $a_i^{\min}$ and $a_i^{\max}$ represent the lower and the upper limits of the component a_i of $\boldsymbol{a}$. The goal in the environment is defined as a sub-region $R_g \in R$. There can be obstacles O_i that are also defined as sub-regions $R_{O_i} \in R$. Obstacles are prohibited regions. The reward function is defined as:

$$R(\boldsymbol{x}) = \begin{cases} 10, \text{ if } \boldsymbol{x} \in R_g; \\ 0, \text{ otherwise.} \end{cases} \tag{15}$$

Figure 2 shows the results of the QFCS and Q-learning on a 1-environment with $x_1^{\min} = 0$, $x_1^{\max} = 10$, $a_1^{\min} = -2$, $a_1^{\max} = 2$ and $R_g = \{x_1 \mid 6 \leq x_1 \leq 7\}$. The given actions allow for left and right movement over the x axis. We can think of this problem as a generalized frog problem, where the frog can jump forward and backward more than one step to catch the fly. This is particularly important because if the fly is too far away, it cannot be reached by the frog in a single step. Variables x_1 and y_1 of QFCS were identified with the 1-environment variables x_1 and a_1. The parameters used by QFCS were $N = 200$, $\theta_{GA} = 1000$, $n = m = 1$, $\delta_0 = 500$ and $250,000$ trials were performed per run. Results show how in this simple problem both systems, QFCS and Q-learning, converge equally fast, but QFCS is a little bit more precise. This is due to the discretization of variables of x_1 and a_1 in Q-learning. This discretization was done with 100 intervals in x_1, and 40 intervals in a_1.

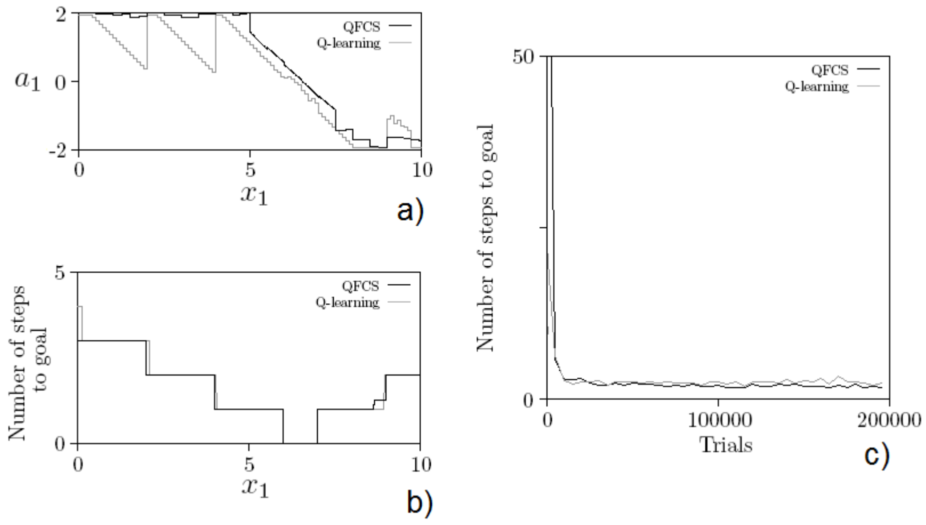


Fig. 2. (a) Average over 20 runs for the learned action. (b) Average over 20 runs of the number of steps to reach goal after learning. (c) Average over 20 runs of the number of steps to reach goal while learning.

Figure 3 shows the classifiers evolved by QFCS and their approximations to the $Q(x_1, a_1)$ function. Each classifier represents a curve in the combined input-output space (x_1, a_1); an approximation to $Q(x_1, a_1)$ is learned by those classifiers only over those curves. These curves were initially equally distributed over all the input-output space. Therefore, QFCS has obtained classifiers, that are over the region, that represent the best actions according to the Q-learning technique defined in Eqn. 2.

QFCS was also tested in two different 2-environment problems. The first one, called *World1,* was defined by $x_i^{\min} = 0$, $x_i^{\max} = 10$, $a_i^{\min} = -2$, $a_i^{\max} = 2$ and

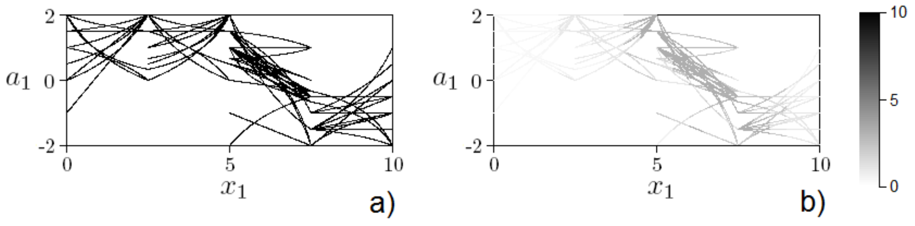


Fig. 3. (a) Classifiers evolved by QFCS. (b) Function $Q(x_1, a_1)$ learned by QFCS in a gray scale over the classifiers evolved

$R_g = \{(x_1, x_2) \mid 3 \leq x_1 \leq 4 \wedge 5 \leq x_2 \leq 6\}$. The second one, called *World2,* was with the same parameters except for $R_g = \{(x_1, x_2)|7 \leq x_1 \leq 8 \wedge 2 \leq x_2 \leq 3\}$ and that it had one obstacle $R_{O_1} = \{(x_1, x_2)|3 \leq x_1 \leq 5 \wedge 0 \leq x_2 \leq 6\}$. Variables x_i and y_i of QFCS were identified with the 2-environment variables x_i and a_i. The parameters used by QFCS were $N = 800$, $\theta_{GA} = 5,000$, $n = m = 2$, $\delta_0 = 1000$ and with a $1,000,000$ trials per run. Q-learning was also done with a discretization of 100×100 in the input space and with 40×40 in the output space but it was observed that this algorithm does not converge after one million trials. Therefore, we reduced the resolution to 25×25 in the input and 10×10 in the output to get convergence. The problem is that, with low resolution, Q-learning loses its accuracy and can learn only suboptimal solutions. Figure 4 shows that QFCS converges at a speed similar to Q-learning and produces similar results. Figure 5 shows the results obtained in the World2.

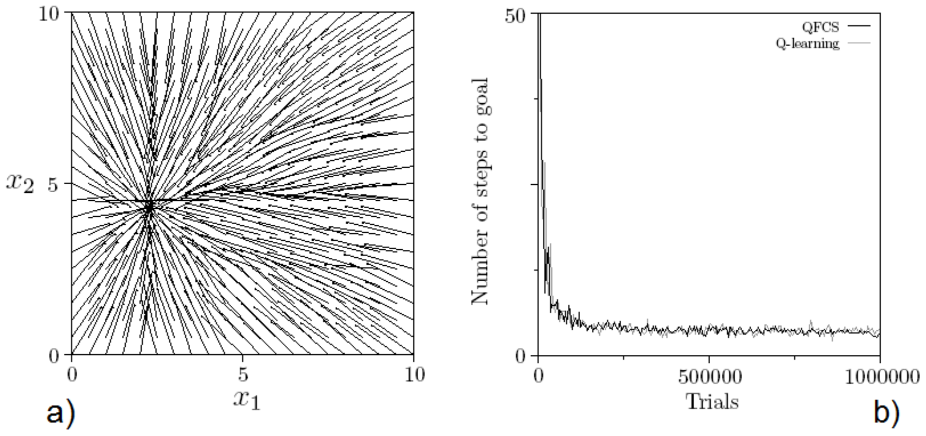


Fig. 4. (a) Average over 20 runs of the action vector field learned. (b) Average over 20 runs of the number of steps to reach goal while learning.

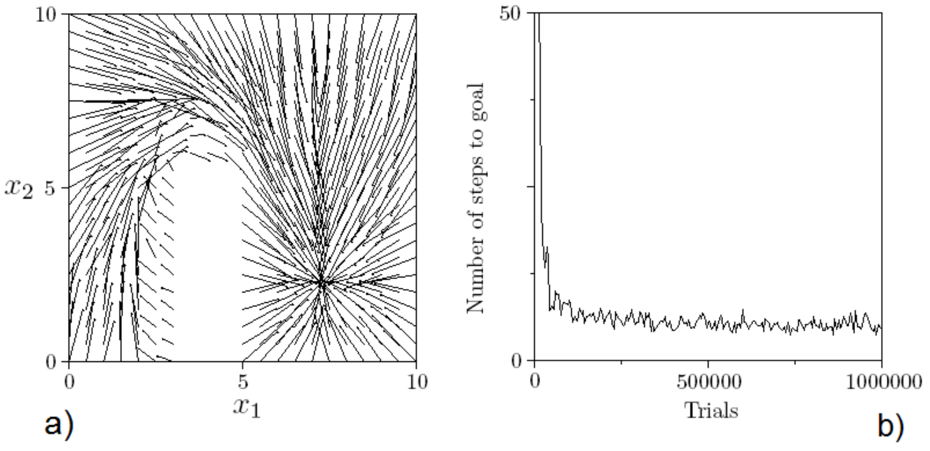


Fig. 5. (a) Average over 20 runs of the action vector field learned. (b) Average over 20 runs of the number of steps to reach goal while learning.

6 Conclusions

We have shown how QFCS can deal with the n-Environment problem defined above. This issue belongs to reinforcement learning problems. QFCS uses the Q-function to determine its actions. In that way, QFCS evolved the classifiers in those parts of the combined input and output space where the Q-function is important to the decision process. Our approach focused on the use of fuzzy logic since, as it has been shown, it is expressive enough for the task. This approach can indeed represent continuous inputs and outputs, as shown in the frog problem, a problem where this feature is more relevant.

Acknowledgments. This research was supported by the Tecnológico de Monterrey (ITESM) under the Research Chair CAT-010.

References

1. Holland, J.H.: Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. Machine Learning: An Artificial Intelligence Aproach 2 (1986)
2. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation 3(2), 1–44 (1994)
3. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, Ribera del Loira, 28. 28042 Madrid (Spain) (2004)
4. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, London (1998)
5. Wilson, S.W.: Get real! XCS with continuous valued inputs. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 1999. LNCS (LNAI), vol. 1813, p. 209. Springer, Heidelberg (2000)

6. Stone, C., Bull, L.: For real! XCS with continuous valued inputs. Evolutionay Computation 11(3), 299–336 (2003)
7. Dam, H.H., Abbass, H.A., Lokan, C.: Be real! XCS with continuous valued inputs. In: Genetic and Evolutionary Computation Conference, pp. 85–87 (2005)
8. Wilson, S.W.: Function approximation with a classifier system. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2001), vol. 4, pp. 974–981 (July 2001)
9. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.: XCS with computable prediction in continuous multistep environments. ILLiGAL Report 2005018, Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign (May 2005)
10. Wilson, S.W.: Three architectures for continuous action. In: Kovacs, T., Llorà, X., Takadama, K., Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2003. LNCS (LNAI), vol. 4399, pp. 239–257. Springer, Heidelberg (2007)
11. Tran, T.H., Cédric Sanza, Y.D., Nguyen, D.T.: Xcsf with computed continuous action. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 1861–1869 (2007)
12. Wang, L.X.: A Course in Fuzzy Systems and Control. Prentice Hall, Upper Saddle River (1996)
13. Valenzuela-Rendón, M.: The fuzzy classifier system: A classifier system for continuosly varying variables. In: Proceedings of the Fourth International Conference in Genetic Algorithms, pp. 346–353 (1991)
14. Valenzuela-Rendón, M.: Reinforcement learning in the fuzzy classifier system. Expert Systems with Applications 14, 237–247 (1998)
15. Parodi, A., Bonelli, P.: A new approach to fuzzy classifier system. In: Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 223–230 (1993)
16. Bonarini, A.: Evolutionary learning of fuzzy rules: Competition and cooperation. Fuzzy Modeling: Paradigms and Practice, 265–284 (1996)
17. Bonarini, A., Bonacina, C., Matteucci, M.: Fuzzy and crisp representations of real-valued input for learning classifier systems. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 1999. LNCS (LNAI), vol. 1813, pp. 228–235. Springer, Heidelberg (2000)
18. Matteucci, M.: Learning fuzzy classifier systems: Architecture and explorations (May (2000), `http://citeseer.ist.psu.edu/matteucci00learning.html`

A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity

Raymond Ros[1] and Nikolaus Hansen[2]

[1] Univ. Paris-Sud, LRI, UMR 8623 / INRIA Saclay, projet TAO, F-91405 Orsay, France
Raymond.Ros@lri.fr
[2] Microsoft Research–INRIA Joint Centre, 28 rue Jean Rostand, 91893 Orsay Cedex, France
Nikolaus.Hansen@inria.fr

Abstract. This paper proposes a simple modification of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for high dimensional objective functions, reducing the internal time and space complexity from quadratic to linear. The covariance matrix is constrained to be diagonal and the resulting algorithm, sep-CMA-ES, samples each coordinate independently. Because the model complexity is reduced, the learning rate for the covariance matrix can be increased. Consequently, on essentially separable functions, sep-CMA-ES significantly outperforms CMA-ES. For dimensions larger than a hundred, even on the non-separable Rosenbrock function, the sep-CMA-ES needs fewer function evaluations than CMA-ES.

1 Introduction

The search space dimensionality, n, plays an essential role in real parameter $\mathbb{R}^n$ optimisation where a non-linear *objective function*, $f : \mathbb{R}^n \to \mathbb{R}$, is to be minimised. Its importance is emphasised by the notion of *curse of dimensionality*: the search space volume increases exponentially with n, making space filling sampling intractable even for moderate dimensionalities. Difficult real parameter optimisation problems also exhibit essential dependencies between the parameters, and learning these dependencies has been successfully addressed by covariance matrix adaptation (CMA) [2,4]. The CMA learns all pair-wise dependencies between all parameters by updating a covariance matrix for the sample distribution. The CMA was originally introduced for evolution strategies (ESs) but recently applied also in Evolutionary Gradient Search [1]. Empirical results indicate that, in order to learn the complete covariance matrix, the *number of objective function evaluations* usually scales sub-quadratically with n [3,4].

In what follows, we will assume a black-box scenario in which function evaluations on f are the only way to gather insights into the nature of f (and therefore to make a reasonable proposal for a solution vector with small function value). The number of function evaluations to reach a target function value is regarded as *search costs*. Furthermore, we call a function f *separable* if the parameters of f are independent in that the global optimum can be obtained by n one-dimensional optimisation procedures along the coordinate axes for any given initial point.

Motivation. A principle limitation of CMA results from the degrees of freedom, $\frac{n^2+n}{2}$, in the covariance matrix, also referred to as strategy parameters. The full learning task scales roughly with n^2 (see *e.g.* [4]) and can dominate the search costs (in this case, the learning phase is much longer than the convergence phase). A second limitation lies in the *internal* computational complexity. (i) Sampling a general multivariate normally distributed random vector has a complexity of n^2 (per sampled n-dimensional vector). A matrix-vector multiplication needs to be conducted. (ii) Updating the covariance matrix has a complexity of $(\mu + 1)n^2$ since the so-called rank-μ update [3] amounts to μ covariance matrix updates. (iii) Factorising the covariance matrix $\boldsymbol{C}$ into $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{C}$ has a complexity of n^3. The factorisation is needed to sample the multivariate normal distribution with covariance matrix $\boldsymbol{C}$. Usually, this computation is postponed until after $n/10$ generations and slightly outdated distributions are sampled [4]. Consequently, the complexity of this step becomes n^2 per generation.[1] In conclusion, several steps in the CMA algorithm have a computational complexity of $\Theta\left(n^2\right)$.

The most obvious option toward improving the scaling behaviour for the search costs is to *reduce the degrees of freedom* in the covariance matrix. We think of several ways to reduce the degrees of freedom, resulting in a family of potentially useful modifications of CMA-ES which *trade off model complexity for learning speed*. As long as the model complexity remains sufficient, search costs decrease because of a reduced learning period. In this paper, we pursue the arguably simplest modification of CMA that reduces the degrees of freedom in the covariance matrix to n. Even though we interpret this modification, sep-CMA, rather as a preliminary step, it reveals some interesting, surprising and promising perspectives on its own.

Previous Works on Favourably Scaling CMA Variants. Some ESs, which were introduced prior to CMA-ES, implement key features of the CMA-ES and scale linearly with the dimension. In [8], a $(1, \lambda)$-ES with cumulation for individual step-size adaptation is proposed.[2] An extension of this derandomised step-size adaptation, denoted AII-ES in [5], combines this individual step-size adaptation with the adaptation of one direction, overall updating $2n$ strategy parameters.

The MVA-ES algorithm [9] adapts one main (mutation) vector. The time complexity of the algorithm is n according to the size of the main vector. The MVA-ES is efficient in the specific case of objective functions with a single preferred mutation direction. In L-CMA-ES [6] a parameter m allows to control the dimensionality of the representation of the mutation distribution. The learning is restrained to $m \leq n$ main components. For the two extremes, if $m = 1$, L-CMA-ES is somewhat similar to MVA-ES and if $m = n$, it is equivalent to the original CMA-ES.

In this paper, we address another subspace of strategy parameters that can be easily identified: the diagonal of the covariance matrix.

Objectives of this Paper. We address two main objectives. (i) Formulating a smallest possible modification of CMA, denoted as sep-CMA, that can learn a scaling of

[1] More precisely, the computation is postponed until after ${c_{\text{cov}}}^{-1} n^{-1}/5$ generations, where the learning rate for the covariance matrix, c_{cov}, equals approximately $2\,n^{-2}$ for small populations. As the learning rate depends on the parent population size, the complexity becomes n^2 per parent vector.

[2] The algorithm is very similar to $(1, \lambda)$-sep-CMA-ES.

variables in linear time. The sep-CMA-ES is, to our knowledge, the first derandomised evolution strategy with linear time complexity that can exploit a large population effectively, just as the CMA-ES. (ii) Comparing the performance of sep-CMA-ES on both separable *and non-separable* functions to CMA-ES and other previously proposed evolutionary algorithms. Surprisingly, sep-CMA-ES will turn out to be advantageous not only on separable, but also on significantly non-separable functions.

The remainder of this paper is organised as follows. Section 2 will introduce sep-CMA-ES, derived from the original CMA-ES. In Section 3, test functions and the test set-up are given. Results from the experiments are presented in Section 4 and provide insights from which conclusions are drawn in the last section.

2 sep-CMA-ES

We begin by presenting the CMA-ES algorithm, introduced in [4]. The $(\mu/\mu_W, \lambda)$-CMA-ES is described in Alg. 1. The description closely follows [2] in using weighted recombination of offspring along with a rank-μ update of the covariance matrix such that a large population size can be exploited.

For sep-CMA-ES, two simple changes are undertaken in the original CMA-ES. (i) The covariance matrix $\boldsymbol{C}$ is in effect constrained to be diagonal, (ii) the learning rate c_{cov} is increased. When the covariance matrix is diagonal, the mutation distribution is sampled independently in the given coordinate system using n individual variances. The only modification in the CMA-ES appears in line 11 which is modified to be:

$$\boldsymbol{D} = \sqrt{\text{diag}(\boldsymbol{C})} \tag{1}$$

where $\text{diag}(\boldsymbol{C})$ is a diagonal matrix with the same diagonal elements as $\boldsymbol{C}$. The matrix $\boldsymbol{B}$ remains $\boldsymbol{I}$ for all iterations.

Because only the diagonal elements of the covariance matrix are utilized, only the diagonal of the covariance matrix must be updated in line 9 and the time complexity of this step becomes linear in n. All other steps in the algorithm become at most linear, because $\boldsymbol{B} = \boldsymbol{I}$ can be removed from the equations.

In contrast to the CMA-ES, the sep-CMA-ES is not rotationally invariant. The degrees of freedom in the covariance matrix reduce from $n + \frac{n^2-n}{2}$ to n, thus the learning rate in line 9 can be increased. Tests on standard functions using different values for the learning rate c_{cov} were done in [10]. For obtaining a similar behaviour than that of CMA-ES when c_{cov} varies, the learning rate for sep-CMA-ES had to be multiplied by $\frac{n+2}{3}$. In all following experiments, $c_{\text{cov}} = \frac{n+2}{3} c_{\text{cov}}^{\text{def}}$ is used for sep-CMA-ES.

3 Test Functions and Methods

Test Functions All test functions are given in Table 1. We introduce the block-rotated ellipsoid function, $f_{\text{blockelli}}^{\beta,m}$. It is the compound of an axis-parallel ellipsoid function with an $n \times n$ matrix $\boldsymbol{Q}$ with m identical orthogonal matrices of size $\frac{n}{m} \times \frac{n}{m}$ along its diagonal (m blocks). If the number of blocks $m = n$, the function is equivalent to the axis-parallel ellipsoid function, for $m = 1$ block, it is equivalent to the rotated ellipsoid

Parameter Setting:

$$\lambda = 4 + \lfloor 3\ln(n) \rfloor,\ \mu = \lfloor \tfrac{\lambda}{2} \rfloor,\ w_i = \tfrac{\ln(\mu+1)-\ln(i)}{\sum_{j=1}^{\mu}\ln(\mu+1)-\ln(j)}\ (i=1,\dots,\mu),\ \mu_{\mathrm{w}} = \tfrac{1}{\sum_{i=1}^{\mu} w_i^2},$$
$$c_\sigma = \tfrac{\mu_{\mathrm{w}}+2}{n+\mu_{\mathrm{w}}+3},\ d_\sigma = 1 + 2\max\left(0, \sqrt{\tfrac{\mu_{\mathrm{w}}-1}{n+1}} - 1\right) + c_\sigma,$$
$$c_{\mathrm{c}} = \tfrac{4}{n+4},\ \mu_{\mathrm{cov}} = \mu_{\mathrm{w}},\ c_{\mathrm{cov}} = c_{\mathrm{cov}}^{\mathrm{def}} = \tfrac{1}{\mu_{\mathrm{cov}}}\tfrac{2}{\left(n+\sqrt{2}\right)^2} + \left(1 - \tfrac{1}{\mu_{\mathrm{cov}}}\right)\min\left(1, \tfrac{2\mu_{\mathrm{cov}}-1}{(n+2)^2+\mu_{\mathrm{cov}}}\right)$$

Initialisation: $g = 0, \boldsymbol{B} = \boldsymbol{I}, \boldsymbol{D} = \boldsymbol{I}, \boldsymbol{p_\sigma} = (0,\dots,0)^T, \boldsymbol{p}_{\mathrm{c}} = (0,\dots,0)^T, \boldsymbol{C} = \boldsymbol{I}$. The initial value of the parameters $\langle \boldsymbol{x} \rangle_{\mathrm{w}} \in \mathbb{R}^n$ and the step-size $\sigma \in \mathbb{R}$ is problem-dependent.
Repeat until a stopping criterion is reached:

```
1.  $g \leftarrow g + 1$
2.  $\boldsymbol{z_i} \sim \mathcal{N}(0, \boldsymbol{I})$ for $i = 1, \dots, \lambda$
3.  $\boldsymbol{x_i} = \langle \boldsymbol{x} \rangle_{\mathrm{w}} + \sigma \boldsymbol{B}\boldsymbol{D}\boldsymbol{z_i}$
4.  $\langle \boldsymbol{x} \rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$ where $\boldsymbol{x}_{i:\lambda}$ denotes the $i$-th best individual out of the $\lambda$
5.  $\langle \boldsymbol{z} \rangle_{\mathrm{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}$ where $\boldsymbol{z}_{i:\lambda}$ denotes the $i$-th best mutation vector
6.  $\boldsymbol{p_\sigma} \leftarrow (1 - c_\sigma)\boldsymbol{p_\sigma} + \sqrt{c_\sigma(2 - c_\sigma)}\sqrt{\mu_{\mathrm{w}}}\boldsymbol{B}\langle \boldsymbol{z} \rangle_{\mathrm{w}}$
7.  $H_\sigma = \begin{cases} 1 & \text{if } \frac{\|\boldsymbol{p_\sigma}\|}{\sqrt{1-(1-c_\sigma)^{2g}}} < (1.4 + \frac{2}{n+1})E(\|\mathcal{N}(0,\boldsymbol{I})\|) \\ 0 & \text{otherwise (stalling the update of } \boldsymbol{p}_{\mathrm{c}} \text{ if } \boldsymbol{p_\sigma} \text{ is large)} \end{cases}$
8.  $\boldsymbol{p}_{\mathrm{c}} \leftarrow (1 - c_{\mathrm{c}})\boldsymbol{p}_{\mathrm{c}} + H_\sigma \sqrt{c_{\mathrm{c}}(2 - c_{\mathrm{c}})}\sqrt{\mu_{\mathrm{w}}}\boldsymbol{B}\boldsymbol{D}\langle \boldsymbol{z} \rangle_{\mathrm{w}}$
9.  $\boldsymbol{C} \leftarrow (1 - c_{\mathrm{cov}})\boldsymbol{C} + \frac{1}{\mu_{\mathrm{cov}}} c_{\mathrm{cov}} \boldsymbol{p}_{\mathrm{c}} (\boldsymbol{p}_{\mathrm{c}})^T$
        $+ c_{\mathrm{cov}}\left(1 - \frac{1}{\mu_{\mathrm{cov}}}\right)\sum_{i=1}^{\mu} w_i \boldsymbol{B}\boldsymbol{D}\boldsymbol{z}_{i:\lambda}(\boldsymbol{B}\boldsymbol{D}\boldsymbol{z}_{i:\lambda})^T$
10. $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p_\sigma}\|}{E(\|\mathcal{N}(0,\boldsymbol{I})\|)} - 1\right)\right)$
11. $[\boldsymbol{B}, \boldsymbol{D}^2] = $ eigendecomposition$(\boldsymbol{C})$
```

Algorithm. 1. The CMA-ES algorithm; $=$ and $\leftarrow$ denote left-hand assignments. Components in $\mathcal{N}(0, \boldsymbol{I}) \in \mathbb{R}^n$ are independent (0,1)-normally distributed. eigendecomposition$(\boldsymbol{C})$ returns normalized, orthogonal eigenvectors as columns of $\boldsymbol{B}$ and the respective eigenvalue square roots as diagonal elements of $\boldsymbol{D}$. To achieve sep-CMA-ES, line 11 is replaced by $\boldsymbol{D}^2 = \mathrm{diag}(\boldsymbol{C})$ according to Eq. (1), and $c_{\mathrm{cov}} = \frac{n+2}{3} c_{\mathrm{cov}}^{\mathrm{def}}$.

function. The hyper-ellipsoid function, $f_{\mathrm{hyperelli}}$, is used in [8] and is biased to more sensitive components than the ellipsoid function. We also use the well-known Rosenbrock function, f_{Rosen}, which is non-convex and non-separable. The rotated Rosenbrock function ($\boldsymbol{Q} \neq \boldsymbol{I}$) is tested as well. The sum-of-different-powers (Diff-Pow) function, $f_{\mathrm{diffpow}}^{\beta}$, is unimodal and separable but reveals increasing differences in the sensitivities when approaching the optimum.

CPU-time Experiments. The total CPU-time for a run with a given number of function evaluations is measured for different problem dimensions. For these experiments we have implemented the sep-CMA-ES and CMA-ES from the `purecmaes.m` Matlab code[3]. In the CMA-ES algorithm the eigendecomposition is postponed until after

[3] `http://www.bionik.tu-berlin.de/user/niko/purecmaes.m`

Table 1. Test functions. We have $\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}$ and the orthogonal $n \times n$ matrix $\boldsymbol{Q}$ is either $\boldsymbol{I}$ for the axis-parallel case or, for the (fully) rotated function, an angle-preserving transformation generated according to [4]. For the block rotated ellipsoid function, $\boldsymbol{Q}$ equals to a block diagonal matrix with an orthogonal $\frac{n}{m} \times \frac{n}{m}$ matrix repeated m times along its diagonal.

Name	Function	f_{target}
Rosenbrock	$f_{\text{Rosen}}(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left(100\,(y_i^2 - y_{i+1})^2 + (y_i - 1)^2\right)$	10^{-9}
Diff-Pow	$f^{\beta}_{\text{diffpow}}(\boldsymbol{x}) = \sum_{i=1}^{n} \lvert y_i \rvert^{2+\beta\frac{i-1}{n-1}}$, $\beta = 10$ as default	10^{-14}
Block-Rotated Ellipsoid	$f^{\beta,m}_{\text{blockelli}}(\boldsymbol{x}) = \sum_{i=1}^{n} \beta^{\frac{i-1}{n-1}} y_i^2$, $\beta = 10^6$ as default	10^{-9}
Hyper-ellipsoid	$f_{\text{hyperelli}}(\boldsymbol{x}) = \sum_{i=1}^{n}(i\,y_i)^2$	10^{-10}

$\alpha(c_{\text{cov}}n)^{-1}$ generations, with α in $\{0, 0.1, 1\}$. The number of function evaluations for the time measurement is 5×10^4 when α is 0.1 or 1 and the dimension is larger than 320, otherwise 10^4, to make sure the eigendecomposition is computed at least ten times. Three trials are done for each algorithm on each dimension. Two population sizes are tested: $\lambda = 4 + \lfloor 3 \ln n \rfloor$ and $\lambda = 2n$. Experiments were performed on a single (no hyper-threading) Intel Core 2 processor 2.66GHz with 2GB RAM.

Performance Experiments. We measure the number of function evaluations to reach the target function value from successful runs. For lower dimension ($n < 100$), 11 runs, otherwise 2 runs are conducted. If the target function value given in Table 1 is reached within 10^7 function evaluations, the run is considered successful. On the Rosenbrock function, at most 30%, usually less, of the runs per set-up converged to the local optimum with CMA-ES or sep-CMA-ES. These unsuccessful runs are disregarded in our performance analysis. The rotation matrix $\boldsymbol{Q}$ is changed for every single run, the same set of rotation matrices is used for testing both algorithms.

We use a Scilab version of the sep-CMA-ES and CMA-ES. For all problems, the starting point $\langle \boldsymbol{x} \rangle_{\text{w}}^{(0)}$ is chosen uniformly in $[-20, 80]^n$ and the initial step-size $\sigma^{(0)} = 100/3$ is one third of the interval width. In addition to the comparison of sep-CMA-ES to CMA-ES, we also compare to previously published results where we use the same starting point, initial step-size (when available) and population sizes as those described in each of the works cited below.

4 Results and Discussion

CPU-time Experiments. Figure 1 displays the total CPU-time divided by the number of function evaluations versus dimension for sep-CMA-ES and CMA-ES. For the default population size (left subfigure), sep-CMA-ES performs much faster. In larger dimensions, the time complexity of sep-CMA-ES empirically scales like $n^{1.2}$, the time complexity of CMA-ES scales like $n^{2.7}$ if the eigendecomposition in CMA-ES is done at each iteration ($\alpha = 0$) and becomes slightly sub-quadratic if outdated covariance matrices are used ($\alpha = 0.1$ and 1), but sep-CMA-ES is still faster by a factor of at least

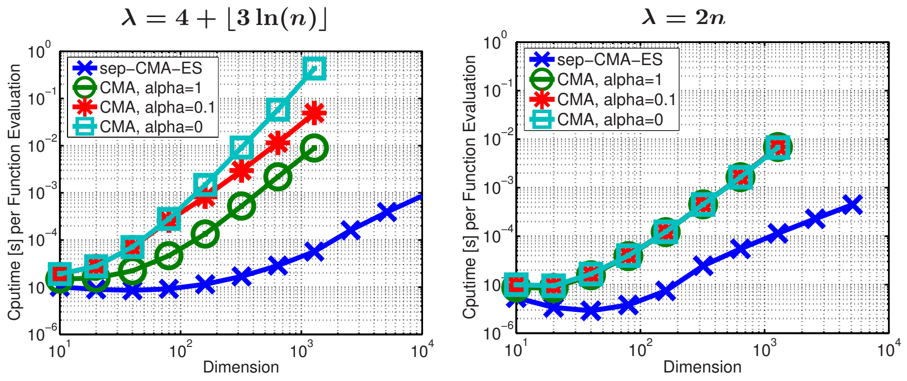


Fig. 1. CPU time per number of function evaluation for sep-CMA-ES (dark) and CMA-ES on the axis-parallel ellipsoid function for two different population sizes λ. The eigendecomposition of the covariance matrix in CMA-ES is postponed until after $(c_{\text{cov}}n)^{-1}\alpha$ generations. Lines show the median of three trials, vertical error-bars show minimum and maximum (all indistinguishable).

six for $n = 100$ and at least a hundred for $n = 1000$. For $\lambda = 2n$ (right subfigure), sep-CMA-ES empirically achieves linear time complexity in larger dimensions whereas the time complexity of CMA-ES is quadratic. Again, sep-CMA-ES is clearly faster than CMA-ES by a factor of ten and forty for $n = 100$ and 1000 respectively.

Performance Experiments. Figure 2 shows the average number of function evaluations to reach f_{target} on different functions. On the separable Diff-Pow and ellipsoid function (left subfigures), the sep-CMA-ES outperforms CMA-ES by a factor of ten in 100-D and the performance gap widens as the dimension increases.

The performance of sep-CMA-ES deteriorates on the rotated functions: on the Diff-Pow function no run reached the target function value. On the block-rotated ellipsoid function, sep-CMA-ES does not succeed with 1 block (*i.e.* rotated ellipsoid function) except for $n = 2$. As the number of blocks increases, the sep-CMA-ES performs gradually better on the block-rotated ellipsoid function. For 2 blocks, sep-CMA-ES outperforms CMA-ES in dimensions larger than 200. Already for 4 blocks, where the condition number within the non-separable sub-problems equals to $10^{6/4} \approx 30$, sep-CMA-ES outperforms CMA-ES in all dimensions.

On the Rosenbrock function (non-rotated) the sep-CMA-ES is roughly 50 times slower than the CMA-ES in small dimensions (cf. right of Fig. 2). With increasing dimension the difference vanishes and, surprisingly, *sep-CMA-ES outperforms CMA-ES on the Rosenbrock function for dimension* $n > 100$ while the success rates remain close for both algorithms. This effect cannot be observed on the rotated Rosenbrock function, where sep-CMA-ES is outperformed by CMA-ES at least by a factor of ten up to 100-D.

The bottom-right of Fig. 2 investigates the advantage of sep-CMA-ES on cheap to evaluate objective functions. By multiplying the number of function evaluations needed on the Rosenbrock function by the CPU-time per function evaluation as given in Fig. 1, we obtain the *overall CPU-time* for optimising the non-separable Rosenbrock function

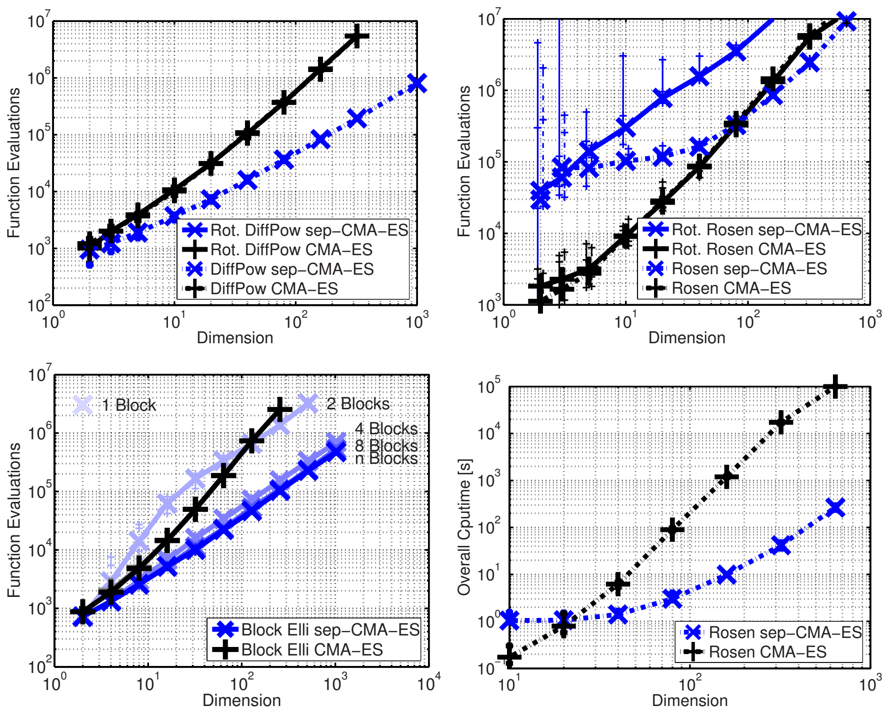


Fig. 2. Experimental results of sep-CMA-ES ($\times$) on the Diff-Pow function (top-left), the block-rotated ellipsoid (bottom-left) and Rosenbrock function (right), compared to CMA-ES (+). The functions are tested in their axis-parallel version (dashed lines) and in their rotated version (plain lines). Lines show median, vertical error-bars show quartiles of the number of evaluations of the successful out of 11 runs on smaller dimension ($n < 100$), out of 2 runs otherwise.

(assuming the CPU-costs of the function evaluations are that of the axis-parallel ellipsoid function as used in the experiments from Fig. 1). The sep-CMA-ES becomes faster for dimensions larger than 20. In 100-D, sep-CMA-ES is already 50 times faster than CMA-ES and the gap widens with increasing dimension. The bottom-right subfigure of Fig. 2 is an optimistic scenario since the axis-parallel ellipsoid function is very cheap. More CPU-expensive functions will narrow the gap and shift the point where the two curves cross to the right, but never beyond $n = 100$.[4]

Comparison to Other Algorithms. Tables 2, 3 and 4 compare the performance of sep-CMA-ES with previous works. On separable functions (f_{elli}, $f_{\text{hyperelli}}$, f_{diffpow}), the sep-CMA-ES performs comparable to indi-ES [8] and AII-ES [5] (Tables 2 and 3) and greatly outperforms MVA-ES [9] and L-CMA-ES [6,7] (Tables 3 and 4), while the latter are rotational invariant. On the Rosenbrock function, AII-ES and L-CMA-ES perform better than sep-CMA-ES, because they can learn a limited number of correlations.

[4] The reason sep-CMA-ES becomes faster is not only because of its smaller time complexity but also because the search costs become lower.

Table 2. Mean number of function evaluations $[\times 10^3]$ to reach the target function value from 3 runs, plus-minus the standard deviation when available, $n = 30$. All functions are used in their non-rotated version. On the hyper-ellipsoid and Diff-Pow function, $\sigma^{(0)} = 1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = (1, ..., 1)^T$. On the Rosenbrock function, $\sigma^{(0)} = 0.1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = \mathbf{0}$.

Function	f_{target}	indi-ES [8]	CMA-ES	sep-CMA-ES	
		(1, 10)-select.	$(7/7_W,\ 14)$	(1, 10)-select.	$(7/7_W,\ 14)$
$f_{\mathrm{hyperelli}}$	10^{-10}	6.6	13±3%	7.2±4%	**5.9±4%**
$f_{\mathrm{diffpow}}^{n-1}$	10^{-20}	9.7	79±4%	19±3%	**9.6±3%**
f_{Rosen}	10^{-6}	80	**45±2%**	81±2%	106±3%

Table 3. Mean number of function evaluations $[\times 10^3]$ to reach given target function value 10^{-9} from 3 runs, plus-minus the standard deviation when available, $n = 20$. In the case of MVA-ES, the range between maximum and minimum from 70 runs is displayed. All functions are used in their non-rotated version. On the ellipsoid function, $\sigma^{(0)} = 1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = (1, \ldots, 1)^T$. On the Rosenbrock function, $\sigma^{(0)} = 0.1$, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)} = \mathbf{0}$. No success was observed for MVA-ES on the ellipsoid function (in 3.5×10^5 function evaluations).

Function	AII-ES [5]	MVA-ES [9]		CMA-ES	sep-CMA-ES	
	(1, 10)-select.	(1, 10)-select.	(5, 35)	$(6/6_W,\ 12)$	(1, 10)-select	$(6/6_W,\ 12)$
f_{elli}	12	no success	no success	20±0.6%	7.6±2%	**5.4±2%**
f_{Rosen}	**21**	57±50%	78±45%	**21±3%**	78±3%	116±1%

Table 4. Mean number of function evaluations $[\times 10^3]$ to reach the target function value 10^{-14} from 3 runs, plus-minus the standard deviation when available, $n = 30$, $\lambda = 14$. All functions are used in their non-rotated version. On the ellipsoid function, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)}$ is chosen randomly in $[-5, 5]^n$, $\sigma^{(0)} = 5$. On the Rosenbrock function, $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)}$ is chosen randomly in $[-2, 2]^n$, $\sigma^{(0)} = 2$.

Function	L-CMA-ES [6,7]		CMA-ES		sep-CMA-ES	
	rank-1, $m = 5$	rank-1, $m = 15$	rank-1	default	rank-1	default
f_{elli}	1900	700	46±0.4%	45±0.8%	**11±7%**	**11±0.5%**
f_{Rosen}	53	63	52±27%	**51±5%**	134±7%	191±2%

5 Summary and Conclusion

We presented the sep-CMA-ES algorithm, a simple modification of the CMA-ES that reduces the $n + \frac{n^2-n}{2}$ degrees of freedom in the covariance matrix of the original algorithm to only n diagonal components, where n is the problem dimension. Consequently, in contrast to the CMA-ES, dependencies between variables are not captured and coordinates are sampled independently. Just like CMA-ES, the sep-CMA-ES can exploit a large population. The **advantages** of sep-CMA-ES are twofold: (i) it reduces the internal time and space complexity of CMA-ES from quadratic to linear. (ii) the learning rate for the covariance matrix can be increased by a factor of about $\frac{n}{3}$, considerably accelerating the adaptation of axis-parallel distribution ellipsoids.

Evaluation Results. We evaluated sep-CMA-ES on separable and non-separable test functions by *measuring numbers of function evaluations*. Fully separable problems mainly served to confirm a proper implementation. On separable functions sep-CMA-ES significantly outperforms CMA-ES and the scale-up with the dimension is linear.

We introduced the **block-ellipsoid**, a convex-quadratic test function for which the "degree of non-separability" can be controlled. For low and moderate degrees of non-separability (relating to "non-separable condition numbers" of up to 30) the sep-CMA-ES outperforms CMA-ES *in all dimensions*, roughly by a factor of $\frac{n}{10} + 1$. For a larger degree of non-separability (relating to a non-separable condition number of 1000) an advantage can be observed only in larger dimension ($n > 100$). On the fully non-separable ellipsoid, CMA-ES is always far superior.

The well-known **Rosenbrock** function exhibits relevant dependencies between the variables posing no principle obstacle for sep-CMA-ES. In low dimensions, sep-CMA-ES is about 50 times slower than CMA-ES. The performance difference diminishes with increasing dimension and for dimensions $n > 100$, *sep-CMA-ES becomes faster than CMA-ES*. This effect can be attributed to the given coordinate system: on the *rotated* Rosenbrock function sep-CMA-ES never outperformed CMA-ES up to dimension 320.

Implications. We perceive two principle benefits from sep-CMA-ES. First, the study of sep-CMA-ES allows to explicitly measure the benefits and drawbacks from learning dependencies. We can quantify the gain or loss that can be attributed to the ability to adapt the complete covariance matrix in CMA-ES. The sep-CMA-ES also allows insightful cross-comparisons with other "separable" algorithms (with only coordinate-wise operations). Second, the application of sep-CMA-ES to **real-world problems** will be advantageous (compared to CMA-ES) on high-dimensional objective functions, which either do not have too intricate dependencies between the decision variables (as it is the case for the Rosenbrock function) or are cheap to evaluate. In the first scenario, sep-CMA-ES needs fewer function evaluations if the adaptation of the scaling of variables helps to solve the function. The second scenario favors sep-CMA-ES, when the strategy internal time complexity becomes relevant, where CMA-ES is roughly $\frac{n}{10} + 1$ times slower. We presented natural examples for both scenarios, where sep-CMA-ES outperforms CMA-ES by a factor of about ten already in 100-D.

For combining the advantages of CMA-ES and sep-CMA-ES in moderate or high dimension, we propose a simple **policy**: using sep-CMA-ES for the first 100 to 200 times n/λ iterations and afterwards CMA-ES retaining the acquired diagonal covariance matrix.[5] The underlying rationale is that the first 100 to 200 times n/λ iterations are almost negligible compared to the adaptation costs for the full covariance matrix afterwards. In some cases, this policy will be adversarial. Using only CMA-ES will be visibly better if most necessary dependencies can be learned quickly with the CMA-ES in the beginning already. Sticking to sep-CMA-ES will be significantly better if the necessary scaling continuously varies (like on $f^{\beta}_{\mathrm{diffpow}}$) and therefore the fast learning of sep-CMA remains beneficial.

[5] Using sep-CMA can improve the sometimes slow progress of CMA-ES in the very beginning of the optimisation which we ascribe partly to non-elitism and partly to the inability of CMA-ES to quickly reduce variances in single coordinates.

Finally, the sep-CMA will serve as a stepping stone to other variants of CMA with linear time and space complexity that we plan to develop.

References

1. Arnold, D., Salomon, R.: Evolutionary gradient search revisited. IEEE Transactions on Evolutionary Computation 11(4), 480–495 (2007)
2. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a new evolutionary computation. Advances on estimation of distribution algorithms, pp. 75–102. Springer, Heidelberg (2006)
3. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation. Evolutionary Computation 11(1), 1–18 (2003)
4. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9(2), 159–195 (2001)
5. Hansen, N., Ostermeier, A., Gawelczyk, A.: On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In: Eshelman, L.J. (ed.) Proceedings of the 6^{th} International Conference on Genetic Algorithms, pp. 57–64. Morgan Kaufmann, San Francisco (1995)
6. Knight, J.N., Lunacek, M.: Reducing the space-time complexity of the CMA-ES. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 658–665. ACM Press, New York (2007)
7. Knight, J.N., Lunacek, M.: Reducing the space-time complexity of the CMA-ES: Addendum. In: Errata for [6] (March 2008), `http://www.cs.colostate.edu/~nate/lcmaes/errata.pdf`
8. Ostermeier, A., Gawelczyk, A., Hansen, N.: Step-size Adaptation Based on Non-local Use of Selection Information. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 189–198. Springer, Heidelberg (1994)
9. Poland, J., Zell, A.: Main vector adaptation: A CMA variant with linear time and space complexity. In: Spector, L., Goodman, E.D., Wu, A., Langdon, W.B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 7–11. Morgan Kaufmann, San Francisco (2001)
10. Ros, R., Hansen, N.: A simple modification in CMA-ES achieving linear time and space complexity. Research Report 6498, INRIA (April 2008), `http://hal.inria.fr/inria-00270901/en`

Evolutionary Algorithms for Dynamic Environments: Prediction Using Linear Regression and Markov Chains

Anabela Simões[1,2] and Ernesto Costa[2]

[1] Department of Informatics and Systems Engineering, Coimbra Polytechnic
[2] Centre of Informatics and Systems of the University of Coimbra
abs@isec.pt, ernesto@dei.uc.pt

Abstract. In this work we investigate the use of prediction mechanisms in Evolutionary Algorithms for dynamic environments. These mechanisms, linear regression and Markov chains, are used to estimate the generation when a change in the environment will occur, and also to predict to which state (or states) the environment may change, respectively. Different types of environmental changes were studied. A memory-based evolutionary algorithm empowered by these two techniques was successfully applied to several instances of the dynamic bit matching problem.

1 Introduction

Evolutionary algorithms (EAs) have been applied successfully to a great variety of stationary optimization problems. However, most real-world applications change over time and some modifications have been introduced in EAs in order to deal with this kind of problems: the use of memory, the maintenance of population's diversity or the use of several populations. See ([1],[2]) for a review.

When the environment is dynamic, in some cases we can try to predict the moment and the pattern of the change. Predicting modifications allows anticipating the sudden decrease in performance of an evolutionary algorithm and improve its adaptability. Our method involves the use of a memory of good past individuals, besides the normal population. That memory interplays with two other modules: one based on linear regression and the other supported by a Markov chain. Linear regression is used to estimate when the next change in the environment will happen; Markov chains are used to model what is known about all possible environments and the transitions among them. The Markov chain is used to predict which new environments will most probably appear in the future. The output of the linear regression module is based on the time of past changes. Once that moment is defined we use the Markov chain model to predict how the new possible environments will look like. Before the predicted moment of change we seek from memory good individuals for these new situations and inject them in the normal population. The main goal of this paper is to investigate the effectiveness of using predictors based on linear regression and Markov chains. We assume that the reader is familiar with the concepts of linear regression and Markov chains. Also, due to lack of space we only present a small part of the results we obtained. The interested reader should consult [3]. The remaining text is organized as follows: section 2 describes related work concerning prediction and anticipation used by EAs in the context of dynamic environments. In section 3 we explain the overall architecture of an EA that utilizes the Markov chain prediction and the linear regression module. In section 4 we present the experimental setup used to test the proposed ideas. Experimental results are summarized in section 5. We conclude with some remarks and ideas for future work.

2 Related Work

Recently, several studies concerning anticipation in changing environments using EAs have been proposed. Branke et al. ([4]) try to understand how the decisions made at one stage influence the problems encountered in the future. Future changes are anticipated by searching not only for good solutions but also for solutions that additionally influence the state of the problem in a positive way. These so-called flexible solutions are easily adjustable to changes in the environment.

Stroud ([5]) used a Kalman-Extended Genetic Algorithm (KGA) in which a Kalman filter is applied to the fitness values associated with the individuals that make up the population. This is used to determine when to generate a new individual, when to re-evaluate an existing individual, and which one to re-evaluate. Van Hemert et al. ([6]) introduced an EA with a meta-learner to estimate at time t how the environment will be at time $t+\Delta$. This approach uses two populations, one that searches the current optimum and another that uses the best individuals in the past to predict the future best value. The prediction about the future is made based on observations from the past using two types of predictors: a perfect predictor and a noisy predictor.

Bosman ([7], [8]) proposed several approaches focused on the importance of using learning and anticipation in online dynamic optimization. These works analyse the influence of time-linkage present in problems such as scheduling and vehicle routing. Bosman propose an algorithmic framework integrating evolutionary computation with machine learning and statistical learning techniques to estimate future situations.

Linear regression was used in [9] to improve local convergence in dynamic problems.

3 System's Overview

In this section we will detail each component of the proposed system, called **PredEA**. The major components of the complete architecture are the following: (1) a standard evolutionary algorithm; (2) memory of past good individuals; (3) Markov chain model module; (4) linear regression module.

The dynamics of the environment is defined off-line: the number of different states, the possible environments, the sequence of environments to use during the simulation and the initial state. We emphasize that all this information is completely **unknown** by the EA, and that the Markov model is constructed during the simulation. As the predictions made by the linear regression module may not be exact, we use a parameter, called Δ, to control the maximum estimated error, measured in terms of generations **before** the actual occurrence of the change. More explicitly, if the predicted value returned by the linear regression module is generation g, at generation $g - \Delta$, the Markov model ([10]) is used to predict the next possible states. At that time individuals from the memory are retrieved and introduced in the population, replacing the worst ones. The selected memory individuals are those who were good solutions in the state(s) that are considered to be the next possible ones by the Markov chain model.

Every time a change actually happens, the probabilities of the transition matrix of the Markov chain are updated accordingly. This includes the case when a new state appears. In that situation, the new state is included in the model and, again, the transition matrix is updated. Notice that, during the earlier stages of the simulation prediction is difficult, because the algorithm needs to experience a learning phase to set up the values of the transition matrix. As we will see, the anticipation based on the introduction of useful information from memory, avoids the decrease of the algorithm's performance. Each component will be explained in detail now.

Evolutionary Algorithm. It's a standard memory-based EA. One population of individuals evolve by means of selection, crossover and mutation and is used to find the best solution for the current environment. The memory population is used to store the best current individual, which we do from time to time. When a change happens or is predicted, the information stored in memory is retrieved and used to help the EA to readapt to the new environment.

Memory. Memory is used to store best individuals of the current population. It starts empty and has a limited size (20 individuals). An individual is stored into memory in two situations: (1) if the environment changed in the meantime and no individual related to this new environment was previously stored; (2) if an individual already exists in memory for the current environment, but it is worst than the current best, the latter individual replaces the former in memory. If memory is full we replace the most similar individual, in terms of Hamming distance, by the current best if it is better ([11]). This way we maximize the capacity of the memory to keep an individual for each different environment. This scheme, called *generational replacing strategy*, was proposed in [12] and proved to be very efficient in memory-based EAs for changing environments. Memory is also used to detect changes in the environment: a change occurs if at least one individual of the memory has its fitness changed.

Markov Chain Module. In our approach, each state of the Markov chain corresponds to a template that represents the global optimum for a certain environment. Initially, a maximum number of different states is defined as well as the possible sequence of states that may occur during the algorithmic process. The initial state is randomly chosen among the existing ones. Again we stress that all this information is *unknown* to the algorithm, which works with a Markov model that it builds dynamically. Ideally, after some generations and environmental changes our algorithm will construct a Markov model identical to the hidden one. From then on, the next state(s) can be correctly predicted making possible the introduction of important information **before** the effective change, allowing the continuous adaptation of the EA to the new conditions. Our algorithm starts with its transition matrix filled with zeros. Each time a transition is detected, say from state i to another state j, the probability values involving state i and all the other states j are changed to take into account the number of times the environment moved from i to j.

Linear Regression Module. Knowing the best moment to start using the predicted information provided by the Markov chain module can improve the adaptation's capabilities of our EA. This moment is computed by calling the Linear Regression Module. The method is simple: the first two changes of the environment are stored after they happen (no prediction can be made yet). Based on these two values, a first approximation of the regression line can be built and the regression module starts providing the predictions about the next possible moment of change. Then, each time a change occurs the regression line is updated.

***PredEA* Pseudocode.** Now that we have described the different components we can present the pseudocode of **PredEA**.

```
PredEA($max$, $markov$, $initial\text{-}state$)
 1  Randomly create initial population
 2  Create empty memory
 3  Create the transition matrix with max states filled with zeros
 4  repeat
 5          Evaluate population
 6          Evaluate memory
 7          if Is time to update memory
 8             then Store the best individual
 9                  Set next time to update memory
10          if An environmental change happens
11             then Store performance measures
12                  Update the linear regression line
13                  Predict g (next-change)
14                  Update the algorithm's Markov transition matrix
15          if g (next-change) is close (as defined by $g - \Delta$)
16             then Predict next state(s) (using EA's Markov model)
17                  Search memory for best individual(s) for that(ese) state(s)
18                  Introduce the selected individual(s) into population
            ▷ Standard EA steps
19          Perform selection, crossover and mutation
20          Define next population
21     until Stop-condition
```

max is the maximum number of states of the Markov chain, *markov* is the Markov model defined off-line, and *initial-state* is the randomly chosen initial state for the Markov model.

4 Experimental Design

Experiments were carried out to compare the **PredEA** with a similar algorithm without prediction capabilities (we will refer this second algorithm as **noPredEA**). The latter algorithm is an EA with the direct memory scheme proposed by [2]. Memory is updated using the same time method, but instead of the replacing strategy used by [2] a generational scheme is used instead: after a change is detected, population and memory are merged and the best $N - M$ individuals are selected as a temporary population to go through crossover and mutation, while the memory remains unaffected (N is the size of population, M is the size of memory). The benchmark used was the *dynamic bit matching problem*: given a binary template, the individual's fitness is the number of bits matching the specified template. A set of different binary templates is generated at the beginning of the run. When a change happens, a different template is chosen from that set.

Experimental Setup. In the Table 4 we summarize the EA's settings used in our experiments.

For each experiment, 30 runs were executed and the number of generations was computed based in 200 environmental changes. The overall performance used to compare the algorithms was the best-of-generation fitness averaged over 30 independent runs, executed with the same random seeds. The results were statistically validated ([3]).

Usually, in papers related with the algorithms' performance on changing environments (e.g. [11], [13], [2]), the measures are saved only after the change is detected **and** some actions had been taken (as the introduction of information from memory). This way, we don't know what really happened to the EA's performance instantly after the change. In this work, the performance measure is saved immediately after a change is detected. This way we can see if the information introduced before the change, based on given predictions, is really useful to the algorithm's adaptability.

The number of different states (templates) used in the experimentation was 3, 5, 10, 20 and 50. The environmental transitions were of two kinds: **deterministic**, i.e. the probability to change to the next state is always 1 (this kind

Table 1. Parameters' settings

EA parameters	**value**
individual's representation	binary
initialization	uniform randomly created
population size	80
memory size	20
crossover	uniform, probability 70%
mutation	flip, probability 1%
parent's selection	tournament, size 2
survivors' selection	generational with elitism of size 1
stop criterion	number of generations necessary for 200 environmental changes
goal	maximize matching with template
Δ	{5,10,25}

of dynamics will be denoted by $P_{ij} = 1$) or **probabilistic**, where, in certain states, the transition can be made to different states (this kind of dynamics will be denoted by $P_{ij} <> 1$). The period was changed in two different ways: periodically, every r generations or according to a fixed pattern. In periodic environments the parameter r was used with four different values: 10, 50, 100 and 200 generations between changes. This type of changes will be called *cyclic-periodic environments*. In the second case, a pattern was set and the moments of change were calculated based on that pattern. Four different patterns were investigated: 5-10-5, 10-20-10 (fast), 50-60-70 (medium) and 100-150-100 (slow). This way the intervals between changes are not always the same, but follow some pattern making prediction possible. This means that we tested **80** different situations. Only partial results will be shown (see [3]).

5 Results

Accuracy of Predicted Values using Linear Regression. When changes occur every r generations, linear regression gives correct predictions, since all the observed values are on the regression line. Using a pattern to generate the periodicity of the change, we may have situations where the predicted values are not precise. In the cases of patterns 5-10-5 and 10-20-10, there is an associated error that slowly decreases over time. In these cases, the Δ constant assumed in our implementation (5 generations), was sufficient to reduce to zero that error, and the insertion of individuals in the population was always made before the change occurs. That is not true for the patterns 50-60-70 and 100-150-100, and we had to use an increased value of Δ to avoid a decrease in performance of our algorithm **PredEA**.

PredEA *versus* noPredEA. Results obtained for cyclic-periodic environments (changing every r generations) are given in Table 2 . The best scores are marked in bold.

We used a paired one-tailed **t-test** at a 0.01 level of significance to compare the two algorithms. The results obtained with **PredEA** were always statistically significantly better than the **noPredEA**. Using prediction to insert information before change happens actually improves the EA's performance. In rapidly changing environments ($r = 10$), the improvements introduced with the anticipation of change are clearly positive. Besides, as the number of different states increases, the **noPredEA**'s performance decreases faster than the **PredEA**'s. Using 50 states the results were inferior since, in some cases, the algorithm has not enough time to complete the 'learning phase'. In these cases, more time of evolution is necessary.

Figure 1 shows the typical behavior of the algorithms in the first 5000 generations, using 10 different states with cyclic ($P_{ij} = 1$) dynamics. **PredEA** has a starting phase where the performance is very unstable with a decrease in fitness every time there is an environmental change. This is the 'learning phase' when the algorithm is building the Markov chain model and its transition matrix.

Table 2. Global Results for cyclic-periodic environments and change period **r**

		Number of States				
r	**Algorithm**	**3**	**5**	**10**	**20**	**50**
10	PredEA ($P_{ij} = 1$)	**98.24**	**97.92**	**97.87**	**97.33**	**94.42**
	PredEA ($P_{ij} <> 1$)	**98.10**	**97.78**	**97.25**	**96.55**	**93.55**
	NoPredEA ($P_{ij} = 1$)	89.41	84.90	80.04	74.87	69.69
	NoPredEA ($P_{ij} <> 1$)	89.64	85.41	80.58	75.40	70.38
50	PredEA ($P_{ij} = 1$)	**99.39**	**99.04**	**98.08**	**96.46**	**91.31**
	PredEA ($P_{ij} <> 1$)	**98.90**	**98.39**	**98.69**	**96.45**	**90.19**
	NoPredEA ($P_{ij} = 1$)	98.72	98.39	97.66	95.40	88.29
	NoPredEA ($P_{ij} <> 1$)	98.71	98.39	97.65	95.76	89.28
100	PredEA ($P_{ij} = 1$)	**99.69**	**99.51**	**99.01**	**98.55**	**95.48**
	PredEA ($P_{ij} <> 1$)	**99.43**	**99.67**	**99.29**	**99.14**	**95.10**
	NoPredEA ($P_{ij} = 1$)	99.38	99.24	98.90	98.29	94.11
	NoPredEA ($P_{ij} <> 1$)	99.37	99.24	98.91	98.29	94.70
200	PredEA ($P_{ij} = 1$)	**99.84**	**99.75**	**99.50**	**99.37**	**97.74**
	PredEA ($P_{ij} <> 1$)	**99.72**	**99.79**	**99.64**	**99.56**	**97.75**
	NoPredEA ($P_{ij} = 1$)	99.69	99.62	99.45	99.15	97.04
	NoPredEA ($P_{ij} <> 1$)	99.69	99.62	99.46	99.15	97.34

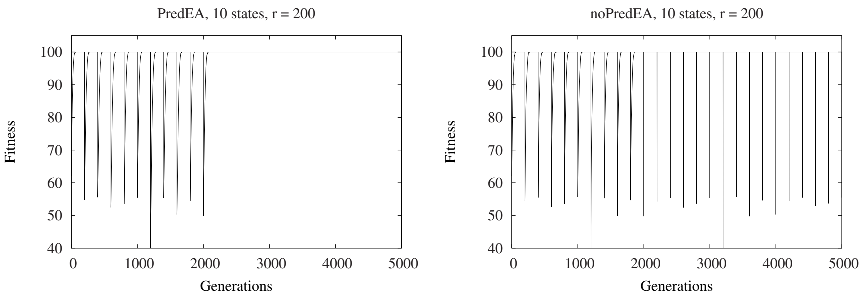


Fig. 1. PredEA versus **NoPredEA**, $r = 200$, 10 states, deterministic

After that initial phase the predictions are correctly made by the two predictor modules and the **PredEA**'s performance reaches an 'equilibrium phase'. On the other hand, the behaviour of **noPredEA** is always very unstable. After a change, we observe a decrease of the fitness and only after retrieving information from memory, which is made immediately after a change happens, the EA recovers. Similar results were obtained for cyclic-pattern environments. For the patterns 50-60-70 and 100-150-100, the value of 5 for Δ constant was not a good choice for the prediction modules. For these two situations, we repeated the experiments adjusting the constant value to 10 and 25, respectively. That way we always anticipated the actual moment of change. The results were better, and since the levels of population's diversity in the two cases are practically the same, this increase in the performance is due to the introduction of retrieved memory information before the change happens. Table 3 shows the global results obtained in all the experiments carried out. Best results are marked with bold. The statistical analysis of the obtained results can be found in [3].

Table 3. Global Results for cyclic-pattern environments 5-10-5, 10-20-10, 50-60-70 and 100-150-100

		Number of States				
Pattern	**Algorithm**	**3**	**5**	**10**	**20**	**50**
5-10-5	PredEAΔ5 ($P_{ij} = 1$)	**97.49**	**97.27**	**97.50**	**97.65**	**95.63**
	PredEAΔ5($P_{ij} <> 1$)	**97.10**	**96.13**	**96.77**	**96.79**	**94.60**
	NoPredEA ($P_{ij} = 1$)	91.98	90.35	85.89	79.36	72.62
	NoPredEA ($P_{ij} <> 1$)	93.18	90.58	86.01	79.90	73.57
10-20-10	PredEAΔ5 ($P_{ij} = 1$)	**98.36**	**98.11**	**97.95**	**97.46**	**94.64**
	PredEAΔ5 ($P_{ij} <> 1$)	**98.24**	**97.49**	**97.12**	**96.29**	**93.01**
	NoPredEA ($P_{ij} = 1$)	91.26	92.25	88.13	80.97	73.40
	NoPredEA ($P_{ij} <> 1$)	94.71	91.52	87.89	81.67	74.51
50-60-70	PredEAΔ5 ($P_{ij} = 1$))	99.54	99.37	98.82	97.04	94.60
	PredEAΔ5 ($P_{ij} <> 1$))	99.52	99.34	98.79	97.24	94.38
	PredEAΔ10 ($P_{ij} = 1$)	**99.80**	**99.65**	**99.31**	**98.60**	**96.52**
	PredEAΔ10 ($P_{ij} <> 1$)	**99.79**	**99.61**	**99.15**	**98.23**	**95.92**
	NoPredEA ($P_{ij} = 1$)	99.44	99.16	98.64	96.45	94.54
	NoPredEA ($P_{ij} <> 1$)	99.44	99.17	98.58	97.02	95.13
100-150-100	PredEAΔ5 ($P_{ij} = 1$)	99.64	99.52	99.20	98.07	95.94
	PredEAΔ5 ($P_{ij} <> 1$)	99.65	99.53	99.23	98.17	96.45
	PredEAΔ25 ($P_{ij} = 1$)	**99.89**	**99.79**	**99.65**	**99.29**	**98.25**
	PredEAΔ25 ($P_{ij} <> 1$)	**99.89**	**99.78**	**99.55**	**99.10**	**97.95**
	NoPredEA ($P_{ij} = 1$)	99.71	99.59	99.27	98.31	97.24
	NoPredEA ($P_{ij} <> 1$)	99.71	99.59	99.29	98.52	97.56

Again, in rapidly changing environments (patterns 5-10-5 and 10-20-10) the incorporation of prediction and the anticipation of change allowed outstanding improvements in the algorithm's performance. **PredEA** also ensures best scores as the number of states increases. In the other two situations, using a suitable value for the Δ constant, **PredEA** also achieves the best results. Figure 2 shows the typical behavior of the algorithms in the first 5000 generations, using 10 different states with cyclic ($P_{ij} = 1$) dynamics. As in the case of cyclic-periodic environments we observe the presence of the learning and equilibrium phases when using **PredEA**. **noPredEA** behaves in the same way as in previous cases. In all situations, except when the **PredEA** has a Δ value of 5, the pattern

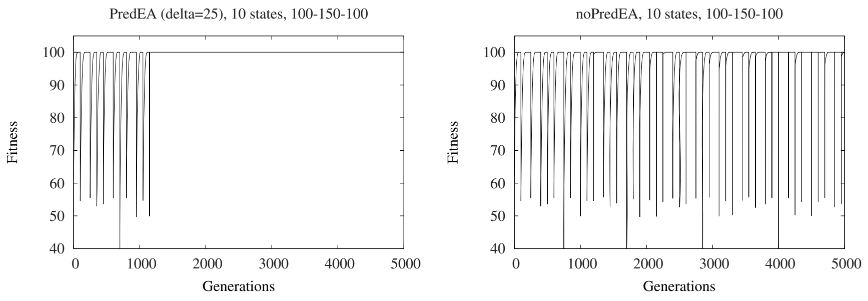


Fig. 2. PredEAversus **NoPredEA**, pattern 100-150-100, 10 states, deterministic

was 100-150-100 and the change deterministic, our algorithm was statistically significantly better than **NoPredEA**.

6 Conclusions and Future Work

We have proposed the integration of prediction capabilities in a standard memory-based evolutionary algorithm to cope with changing environments. Two additional modules were used: one based on linear regression to predict when next change will occur, the other uses a Markov chain which stores the environmental information and is used to predict the next possible state(s).

Analyzing the obtained results, some conclusions can be stated: first, anticipating the moment of change and using information gathered in the past to prepare the algorithm for future environmental changes, significantly improves the EA's adaptability. Second, these improvements have more impact when the environment changes faster. In these cases, if the prediction capabilities are removed, the algorithm has a very poor performance. Third, **PredEA** is more robust than **noPredEA** as the number of repeated states increases. Fourth, the linear regression method, used to predict the moments of subsequent changes, is suitable only for a restricted kind of changing periods. In fact, if there is an error because the effective change occurred before the predicted one, the algorithm's performance is compromised. This is due to an untimely use of the solution obtained from the Markov chain module, making the use of Markov chains predictions unhelpful. Finally, the use of a Markov chain to store the environmental information proved to be a powerful mechanism to keep the history of the changing dynamics which allows the algorithm to learn and predict which states can appear in the next step.

The major limitations of the proposed architecture are related to the linear regression module. First, the use of linear regression to predict future change points is feasible only for certain patterns. For more complex patterns, linear regression may fail, due to large prediction errors. Second, the use of a fixed value for the error interval (the Δ parameter) assumed in the linear regression predicted values is not always effective. If an unsuitable value is used for this constant, the algorithm's performance considerably decreases. Some enhancements are being introduced to improve this module: the use of **non linear regression** and the **dynamic adjustment** of the Δ constant during the simulation. Other landscapes are also being used to test the proposed ideas.

Acknowledgements

The work of the first author was partially financed by the PhD Grant BD/39293/2006 of the Foundation for Science and Technology of the Portuguese Ministry of Science and Technology and High Education.

References

1. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments: a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
2. Yang, S.: Explicit memory schemes for evolutionary algorithms in dynamic environments. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments, pp. 3–28. Springer, Heidelberg (2007)
3. Simões, A., Costa, E.: Evolutionary algorithms for dynamic environments: prediction using linear regression and markov chains. Technical Report TR 2008/01, CISUC (2008)
4. Branke, J., Mattfeld, D.: Anticipation in dynamic optimization: The scheduling case. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., Schwefel, H.P. (eds.) PPSN VI 2000. LNCS, vol. 1917, pp. 253–262. Springer, Heidelberg (2000)
5. Stroud, P.D.: Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations. IEEE Transactions on Evolutionary Computation 5(1), 66–77 (2001)
6. van Hemert, J., Hoyweghen, C.V., Lukshandl, E., Verbeeck, K.: A futurist approach to dynamic environments. In: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp. 35–38 (2001)
7. Bosman, P.: Learning, anticipation and time-deception in evolutionary online dynamic optimization. In: Yang, S., Branke, J. (eds.) GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization (2005)
8. Bosman, P.A.N.: Learning and anticipation in online dynamic optimization. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. Springer, Heidelberg (2007)
9. Bird, S., Xiaodong, L.: Using regression to improve local convergence. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), pp. 592–599. IEEE Press, Los Alamitos (2007)
10. Norris, J.R.: Markov Chains. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge (1997)
11. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: IEEE Congress on Evolutionary Computation (CEC 1999), pp. 1875–1882. IEEE Press, Los Alamitos (1999)
12. Simões, A., Costa, E.: Improving memory's usage in evolutionary algorithms for changing environments. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007). IEEE Press, Los Alamitos (2007)
13. Simões, A., Costa, E.: An immune system-based genetic algorithm to deal with dynamic environments: Diversity and memory. In: Proc. of the 6th Int. Conf. on Artificial Neural Networks, pp. 168–174. Springer, Heidelberg (2003)

Combination of Natural and Numerical Optimization Methods at the Example of an Internal Gas Turbine Cooling Channel

H. Steinbrück, S. Zehner, B. Weigand, and S.O. Neumann

Universität Stuttgart, Institut für Thermodynamik der Luft– und Raumfahrt, Pfaffenwaldring 31, 70569 Stuttgart

Abstract. Iceformation phenomena can be observed in many natural and technical processes. A naturally grown ice layer aspires in steady state to a minimum of energy dissipation. Driven by this goal, this phenomena can be used to optimize complex geometric configurations in a natural manner. But since this state of minimum energy dissipation is seldom the desired goal function in technical applications, this natural experimental optimization method is combined with a subsequent classical numerical optimization using evolutionary algorithms at the example of an internal cooling channel of a gas turbine blade.

1 Introduction

Optimization of complex geometries exposed to fluid flow is often challenging, since local separation and re-attachment phenomena induce local vortices and highly three dimensional flow characteristics. Each change of the geometry influences the flow characteristics, often in an unpredictable way.

For classical numerical optimization, the first choice to be made is the one for the design variables that describe accurately the problem. According to the description in [2], the next step is to formulate the constraints, the objective function and the variable bounds. But this preliminary definition of restrictive parameters, such as the number of the design variables, their position and the entire design space, unfortunately limits the final result of the optimal geometry. Since all classical numerical optimization methods are characterized by the steps "selection" and "variation" of the initial solution [11], the final optimum can only be an offspring of that.

Taking as an example the traditional turbine blade optimization in turbomachinery, the geometry of the blade is described sufficiently by a definite number of characteristic aerodynamic and mechanical parameters such as the inlet and outlet flow angle, trailing and leading edge radius, chord length, etc. [13]. Though the aerodynamic properties of the blade vary significantly in the optimized result, the geometric shape remains always similar due to the restrictive parameterization which is of course desired in this specific case.

However, if the general shape of the optimized geometry is not known a priori or innovative shapes shall be found, an infinite amount of geometric parameters is necessary in order to map all possible morphologies of a geometry. This would require then an unaffordable amount of computational power and time to optimize such a system of indefinite variables.

The approach which is presented here shows instead a novel way of optimizing a geometry according to the surrounding fluid flow conditions using a combination of the natural iceformation method and a subsequently applied numerical optimization. The goal function for the natural process, i.e. the iceformation process, is minimized energy dissipation [9], but this optimum represents only a compromise between minimzed aerodynamic (hydrodynamic) drag and a minimum in heat transfer. Since in technical applications it is often desired to minimize the pressure loss or the heat loss, this naturally pre-optimized geometry is taken in a next step as starting configuration for a further numerical optimization with the objective of minimized drag using evolutionary algorithms. For this, the experimentally determined and parameterized ice contour defines the definite number of design variables and the possible design space and can therefore give an idea of how the optimized contour could look like, while at the same time restrictive human control parameters are limited.

2 Natural and Numerical Optimization Methods

2.1 The Iceformation Method

The growth of an ice layer is a natural process that can be observed in many natural and technical processes. Belonging to the natural optimization methods means for the iceformation method that external human control which might influence the process restrictively, is reduced to a minimum. The ice layer growth is rather based on a combination of heat transfer and momentum transfer phenomena and their interaction with the wall, the latter consisting of the frozen ice layer. The final ice contour is then determined at steady-state by a local heat transfer equilibrium.

In the present study, the method is applied to optimize the shape of the separating web in a channel with a 180° bend as it is characteristic for the internal cooling channels in gas turbine blades. Figure 1 shows a schematic sketch of such a channel configuration as it was used for the investigated ice layer formation.

Hereby, the parent surface - represented by the separating web - is cooled to a temperature below the freezing temperature of the circulating fluid. Due to this cooling the parent surface is covered with an ice layer whose shape is adapted to the fluid flow conditions. Restrictions are only made by choosing the degree of cooling of the parent surface, the flow velocity and the channel geometry. The boundary constraints in detail are:

T_F	... freezing temperature of the fluid (at 0 °C)
T_0, $\bar{u}_0$	... inlet temperature and mean inlet velocity of the fluid
T_W	... wall temperature of the cooled parent surface
W	... channel width
W_{el}	... tip-wall to web distance

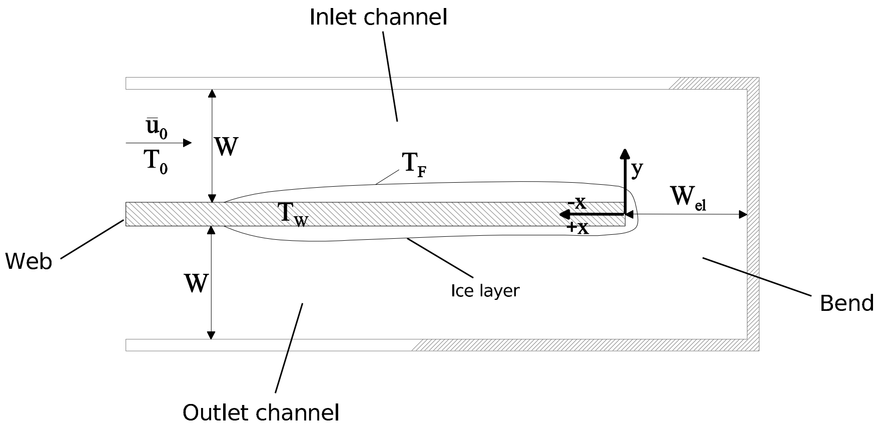


Fig. 1. Schematic sketch of the boundary conditions in the channel with the 180° bend and its ice covered separating web

The circulating fluid and the ice contour are in a state of mutual interaction. The contour then tends to a state of minimum energy dissipation. Since for technical applications often a different goal function such as minimized pressure loss is more important, the contour is taken as starting configuration for a subsequent numerical optimization with a different objective function. But due to the fact that the growth of an ice layer is a natural process, the method gives a hint of how good candidates of web geometries look like and can therefore help to enlarge the pool of possible solutions.

A detailed description of the experimental setup as well as a first numerical analysis of the channel geometries and a comparison of experimental and numerical results can be found in [17] and [18]. In these papers basic experimental and numerical investigations on the change of the flow field caused by ice-formation was conducted and the 2D channel configuration was optimized numerically. Due to the 2D computation secondary flow effects like the Dean Vortices in the turn are neglected. In the present paper however, the focus lies more on the optimization algorithms and a result for a 3D channel configuration is presented since this setup reflects more the experimental conditions.

2.2 Parameterization of the Starting Configuration for the Numerical Optimization Process

Once in the experiment, the contour reaches steady-state, discrete points of the ice layer are measured with a 2D-optical method. In order to use the contour as a starting configuration for a numerical optimization, the measured points must be converted into geometry defining points. This parameterization of the geometry is done using a Bezier-curve approach. Bezier-curves are defined by a control polygon whose points are called Bezier-points. The number of Bezier-points controls the degree of the according polynomial curve. The first and the

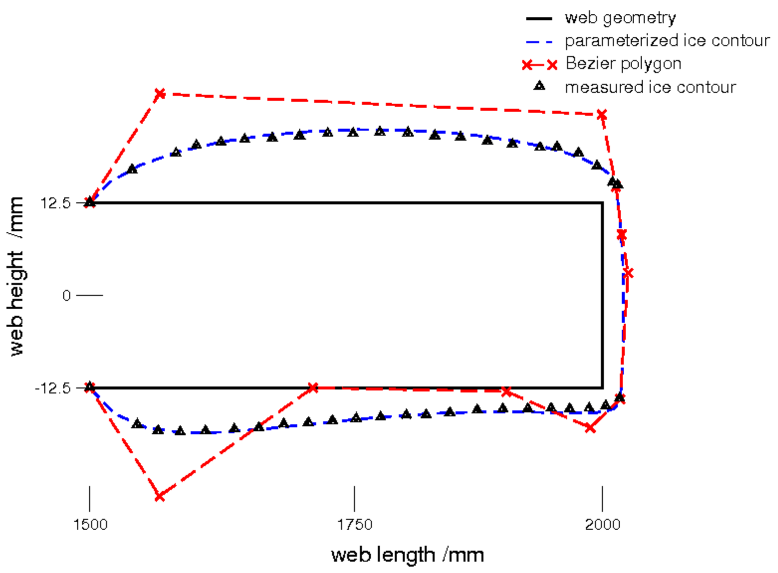


Fig. 2. Bezier points to determine the ice contour on the web

last point of polygon and curve are identical and the straight line between the first two and the last two points of the polygon are tangential to the curve. It is characteristic for this approach, that a change in the coordinates of one single point in the polygon affects the whole curve band but leads always to a smooth, continuous and derivable curve. In that way, wavy contours during the optimization process that might lead to highly selective and not converging individuals can be avoided. This approach was also formerly successfully used to model airfoil optimization, e.g. by Selmin [15] or Sommerer [16].

Discrete points of the Bezier-curve are determined by applying the de Casteljau algorithm [5] while the number of these points can be chosen arbitrarily. Figure 2 shows exemplarily the parameterized mapped ice contoured web that was measured for one configuration of fluid flow velocity and cooling temperature. The according twelve Bezier-points form the surrounding polygon and therefore define the contouring curve, consisting of finally 60 points, which are connected by splines.

For the presented geometry, the Bezier-curve approach leads then to an optimization problem with 12 design variables, since the polygon points are varied in y-direction but kept fixed in x-direction. Their limits are chosen such that the resulting curve is not exceeding the channel's geometric constraint.

Once parameterized, a grid is generated and the geometry is analyzed numerically. Momentum and energy equation are solved using FLUENT$^{\text{TM}}$ 6.3.26 [6]. The geometry is classified according to its "fitness", represented by the pressure drop across the channel. The latter is determined by the difference of the averaged static pressure between inlet and outlet of the channel: $\Delta p = p_{in} - p_{out}$. The optimizer then creates a new set of parameters that represents a new web

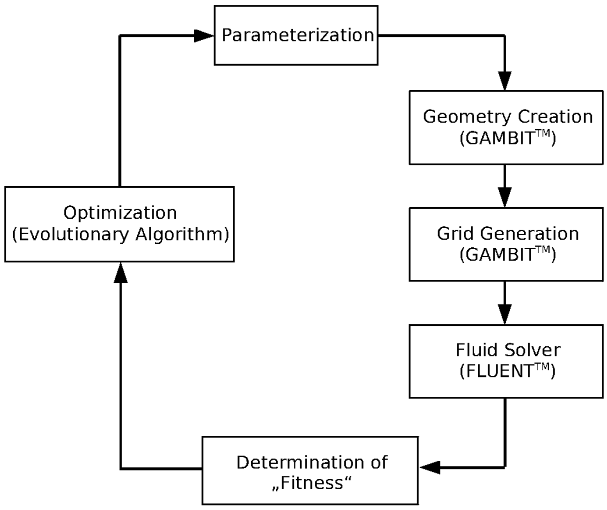

Parameterization
Geometry Creation (GAMBIT™)
Grid Generation (GAMBIT™)
Fluid Solver (FLUENT™)
Determination of „Fitness"
Optimization (Evolutionary Algorithm)

Fig. 3. Optimization procedure for one individual

geometry. This channel geometry is then again analyzed and classified. The procedure for one iteration is depicted in figure 3.

The links between all used programs were steered by a higher ranked shell script. The intersect supply of the parameters was done using FLUENT$^{\text{TM}}$'s journaling features. One run in this process represents one iteration and the process was stopped, when a predefined number of iterations was reached or the pressure drop converged to a minimum.

3 Optimization Algorithms

For the presented problem, Evolutionary Algorithms are especially suitable since the objective function can not be described analytically by a definite function but is the result of a numerical simulation. The algorithm's stochastic search in the design space makes them capable of finding the global optimum in problems with various local optima [3] and represent therefore an appropriate choice.

For this specific application, a combination of a real-coded Genetic Algorithm (GA) from Deb [4] and a commercially available Evolution Strategy (ES) from CAOne$^{\text{TM}}$ is chosen in order to combine the advantages of both algorithms to a hybridized approach [14]. As Hoffmeister and Bäck stated in a comparison of GA and ES [7], both implement the principles "population", "mutation", "recombination" and "selection". The GA however, sometimes tends to converge to premature solutions while the ES, not suffering so significantly from this problem, converges much slower than the GA. This phenomena also occurred in the presented problem and can be seen in figure 4, where the convergence for both algorithms, applied to the present problem of the minimization of the pressure drop in a 2D internal gas turbine cooling channel, is depicted.

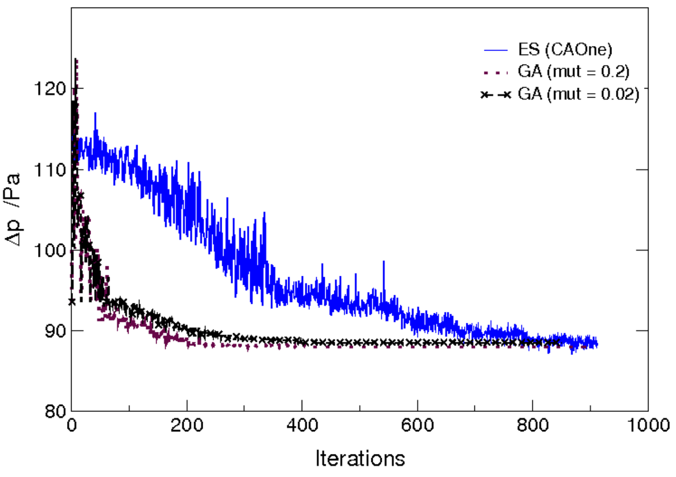


Fig. 4. Convergence of the GA at different mutation rates and the ES

For the parameter study of the computated channel in 2D, the mutation rate for the real coded GA was set from 0.02 to 0.2 and the crossover rate constant to 0.8. The crossover probability for the Evolution Strategy was set to 1/3. Its mutation rate was defined by a decreasing step width, i.e. the higher the iteration, the closer the mutated offspring was positioned in the vicinity of the parent solution.

The recombination method of the ES is done using a $(\lambda, \mu)-$approach. This means that from each of the parents $\mu = 4$, there are $\lambda = 4$ children created. The population size for both algorithms was set to 16. A further increase of the population size had no significant positive influence on the convergence behaviour but was increasing the necessary CPU time.

For these settings, figure 4 shows the variation of the calculated pressure drop across the channel for each of the around 900 simulated 2D web geometries. It can be observed that the GA converges significantly faster than the ES, even more so for the higher mutation rate, but not necessarily to the best solution. For both mutation rates of the GA, the ES – though slower converging – finds a better solution on the long run.

In order to reach still the best solution but with a lower amount of iterations, both algorithms were combined to a hybrid approach in the following way: The first thirteen generations, which correspond to the first 208 iterations, are calculated using the Genetic Algorithm. The best solution of these iterations is then taken as starting configuration for the ES, using the same setting as presented. Applying this combination, the necessary amount of iterations can be decreased to around 600 while maintaining the solution's quality, as shown in figure 5.

The reason why the two algorithms can be combined in that way lies in the different treatment of genetic information at the algorithms [7]. The concept of "population" means for the GA that for a certain number individuals in

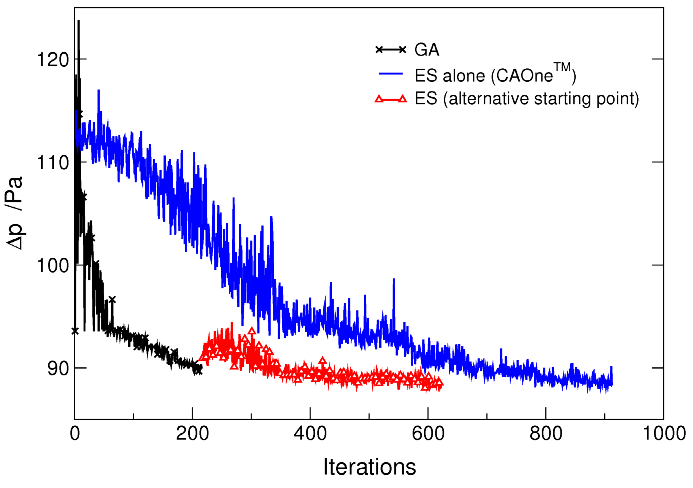


Fig. 5. Convergence of GA and ES for a combined approach

the whole design space are created randomly. For the ES however, this only defines how many children per parent are created by recombination and mainly mutation. As a consequence, for the ES a real starting parent configuration can be chosen from which the offspring approaches in each iteration a better solution, whereas while using the GA, the starting configuration is determined by the chosen design space.

4 Results and Discussion

Figure 6 shows the starting contour and the optimized web contour after 624 iterations for the previously described optimization approach but for a 3D channel geometry. The starting configuration represents the parameterized ice contoured web that had shown in the experiments a reduction in pressure drop of 15% compared to the initial sharp edged web. The optimized web contour reaches finally a calculated pressure drop reduction in the bend of further 17% compared to the starting configuration with an ice contoured web.

The main reason for the decreased pressure drop in the channel is the reduced size of the separation bubble behind the bend. This flow recirculation is mainly responsible for the losses according to Metzger et al. [12]. As a consequence, the optimized contour has filled up the region of recirculation and therefore reduced the losses caused by the flow separation behind the bend.

The optimized web geometry differs slightly from the one that was presented in [17] where only a web contour in 2D was optimized. In the present case of a 3D channel optimization also losses caused by secondary flows in the channel are taken into account. The resulting geometry is also minimizing these and not only the separation bubble behind the bend.

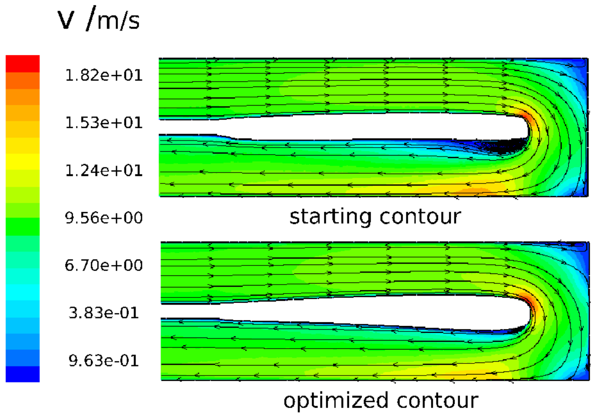


Fig. 6. Velocity plot in the midspan of the starting ice-shaped web and the optimized 3D web contour

Generally, the result of the round web shape with a large radius is not surprising since previous investigations of Idelchik [8] or Metzger et al. [12] indicated that these two factors have the most influence on the pressure drop. Therefore, an experienced engineer would have probably almost guessed that the web tip must have a round shape in order to minimize the pressure drop.

The presented approach here shows the following though: Firstly, the determination of the number of design variables by the parameterization of the ice contour, secondly, starting the numerical optimization with an already good candidate and thirdly the presented combination of evolutionary algorithms, all this together provides the possibility of a fast optimization process while at the same time restrictive human control is reduced to a minimum.

Summarizing, it can be said that the application of the natural iceformation method before a classical numerical optimization can help to provide innovative geometries and create larger manifolds in the pool of possible solutions. Combining the natural and numerical method as presented, leads to a web geometry that performs far better in terms of pressure drop than the initial geometry. This result also showed that the method has proven to be applicable in complex flow conditions and can therefore also be applied at geometries where the optimum has a highly three-dimensional and complicated shape. Further more, the used algorithms are also applicable for multi-objective problems. That means that the problem can be extended such, that not only pressure drop but also heat transfer rates are taken into account.

Acknowledgement

The authors kindly acknowledge the financial support of this project by the DFG (German Science Foundation) and the FVV (Research Association for Combustion Engines).

References

1. CAOtec Software GmbH: CAOne^{TM}, Version 3.1.3, `http://www.caotec.com`
2. Deb, K.: Optimization for Engineering Design – Algorithms and Examples. Prentice-Hall of India Private, Limited, New Delhi (1995)
3. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley and Sons LTD, Chichester (2002)
4. Deb, K. (2006), `http://www.iitk.ac.in/kangal/`
5. Farin, G.E.: Kurven und Flächen im Computer Aided Design, Vieweg (1993)
6. FLUENT Inc., Fluent 6.1 User's Guide Volume 2 (2003)
7. Hoffmeister, F., Bäck, T.: Genetic Algorithms and Evolution Strategies: Similarities and Differences; Parallel Problem Solving from Nature. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 455–466. Springer, Heidelberg (1991)
8. Idelchik, I.E.: Handbook of Hydraulic Resistance, 3rd edn. Begell House, New York (1996)
9. LaFleur, R.S.: Evolution theory for optimal control of a Couette iceform design model. Int. J. Heat Mass. Transfer 35(10), 2617–2629 (1992)
10. LaFleur, R.S., Langston, L.S.: Drag Reduction of a Cylinder/Endwall Junction Using the Iceformation Method. Journal of Fluids Engineering 115, 26–32 (1993)
11. May, R.M.: Exploration of Ecological Systems. Scientific American 239(3), 160–175 (1978)
12. Metzger, D.E., Plevich, C.W., Fan, C.S.: Pressure loss through sharp 180-deg turns in smooth rectangular channels. Transactions of the ASME, Journal of Engineering for Gas Turbines and Power 106, 677–681 (1984)
13. Pierret, S., Van den Braembussche, R.A.: Turbomachinery Blade Design Using a Navier-Stokes Solver and Artificial Neural Network. Journal of Turbomachinery 121, 326–332 (1999)
14. Sastry, K., Goldberg, D., Kendall, G.: Genetic Algorithms. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies, Introductory Tutorials in Optimization and Decision Support Techniques. Springer, Heidelberg (2005)
15. Selmin, V.: Direct Optimization of Transonic Airfoils. In: Periaux, J., Bugeda, G., Chaviaropolulos, P.K., Labrujere, T., Stoufflet, B. (eds.) EUROPT – A European Initiative on Optimum Design Methods in Aerodynamics, Vieweg (1997)
16. Sommerer, A.: Numerische Optimierung adaptiver transsonischer Profile; Dissertation, Institute of Aerodynamics and Gasdynamics, Universität Stuttgart (2005)
17. Steinbrück, H., Zehner, S., Weigand, B., Neumann, S.O., Gier, J.: Optimization of the Web Geometry of a 180-Degree Bend Using an Experimentally Determined Ice Layer as Starting Contour: Part 2 – Numerical Simulation and Optimization. In: Proceedings of ISROMAC12, Honolulu, USA ISROMAC12-2008-20053 (2008)
18. Zehner, S., Steinbrück, H., Weigand, B., Neumann, S.O., Gier, J.: Optimization of the Web Geometry of a 180-Degree Bend Using an Experimentally Determined Ice Layer as Starting Contour: Part 1 – Experimental Results. In: Proceedings of ISROMAC12, Honolulu, USA SROMAC12-2008-20049 (2008)

When Does Quasi-random Work?

Olivier Teytaud

TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud), Bât 490
Univ. Paris-Sud 91405 Orsay, France
teytaud@lri.fr

Abstract. [10,22] presented various ways for introducing quasi-random numbers or derandomization in evolution strategies, with in some cases some spectacular claims on the fact that the proposed technique was always and for all criteria better than standard mutations. We here focus on the quasi-random trick and see to which extent this technique is efficient, by an in-depth analysis including convergence rates, local minima, plateaus, non-asymptotic behavior and noise. We conclude to the very stable, efficient and straightforward applicability of quasi-random numbers in continuous evolutionary algorithms.

Keywords: Evolution Strategies; Derandomization.

1 Introduction

Whereas pseudo-random numbers are supposed to be as close as possible to pure random numbers, quasi-random numbers are designed in order to be more "uniformly" distributed than pure random numbers. Various criteria of uniformity have been developed [9,14]. In some cases, a good value for a criterion ensures a good behavior: for example, a good discrepancy implies a small integration error for numerical integration of functions with finite total variation [8] through Koksma-Hlawka's inequality. Sequences with good properties for these criteria are mainly algebraic (quickly generated) methods [3,14,16,19,23,25]. As a consequence of this important part of science, we can use fast and reliable generators of points, with good uniformity properties. Thanks to scrambling and other tricks (including randomization [11]) [1,2,5,6,12,13,15,17,20,24,26,27,28,29], quasi-random points can be used also in high dimensionality with positive results [18].

Quasi-random numbers have been a revolution basically in numerical integration [14], but it was also used in other areas: quasi-random search [14], path planning [23], active learning [4], approximate dynamic programming [21].

[22], inspired by [10,14], has proposed the use of quasi-random numbers for mutations in the continuous domain. The main results in [22] are about the convergence rate of CMA [7], modified by such an algorithm. However, many questions arised, about the generality of the results: as many people use random numbers as a secure tool for exploration, it is not intuitive that random numbers can be removed from evolutionary algorithms without strong drawbacks

induced somewhere as a counterpart. We here provide an in-depth analysis of quasi-random mutations, through robustness, local minima, needle functions, noise, quality of the quasi-random number generator, and non-asymptotic properties. We conclude to the very wide applicability of quasi-random numbers in continuous evolution strategies.

In all the paper, we use the fitness functions presented in the Matlab/Octave implementation of CMA. When averages and standard deviations are presented, number between "(.)" are median values. Bold font indicate significant improvements, for rank-based tests (Wilcoxon statistics). The result of each algorithm is the best individual of the last generation.

The use of quasi-random mutations is quite straightforward. If your mutation operator is in Algorithm 1, then you just replace it by Algorithm 2.

Algorithm 1. Standard mutation in the continuous domain, with σ a step size and A a linear transformation.

```
Function $x' = Mutation(x, \sigma, A)$
return $x' = x + \sigma A.\mathcal{N}$
```

Algorithm 2. Quasi-random mutation in the continuous domain, with σ a step size and A a linear transformation.

```
Function $x' = Mutation(x, \sigma, A)$
return $x' = x + \sigma A.\mathcal{N}_{qr}$
```

The only difference is the replacement of $\mathcal{N}$, a standard multivariate Gaussian variable, by $\mathcal{N}_{\amalg\nabla}$, its quasi-random counterpart. If $\mathcal{N}$ is generated as in Algo. 3, then you just have to replace it by Algo. 4 in your favorite program.

Algorithm 3. Usual algorithm for generating a standard multivariate Gaussian.

```
Function $\mathcal{N} = MultivariateStandardGaussian(dimension\ d)$
for $i \in \{1, 2, \ldots, d\}$ do
  $x_i = random$ (uniform in $[0, 1]$)
end for
for $i \in \{1, 2, \ldots, d\}$ do
  $\mathcal{N}_i = inverseGaussianPDF(x_i)$
end for
```

In the rest of this paper, DCMA is the standard CMA (covariance matrix adaptation) algorithm, with the transformation above (from Algo. 3 to Algo. 4).

2 Quasi-random Mutations Need Good Quasi-random Sequences

We compare a simple Halton sequence (without scrambling) and Sobol's sequence. It is known that Sobol sequence is much better, in particular for large dimensional problems, but the comparison is particularly impressive in the case

Algorithm 4. Algorithm for generating quasi-random Gaussian numbers. Finally, only one simple loop is replaced by one call to a standard function.

```
Function $\mathcal{N} = MultivariateStandardGaussian(dimension\ d)$
Apply $x = Sobol(d)$.
for $i \in \{1, 2, \ldots, d\}$ do
  $\mathcal{N}_i = inverseGaussianPDF(x_i)$
end for
```

of mutations of evolution strategies. Table 1 presents the comparison between the normalized (see caption) log of the smallest fitness value found by the algorithm in the case of Halton sequence compared to the random classical points and Sobol sequence for 10 function evaluations in dimension 4 and 40 function evaluations in dimension 16.

The explanation is clear on a typical plot of a random walk. Figure 1 (left) shows the sum of quasi-random Gaussian numbers generated with the 3rd and

Table 1. $d \times \log(fitness)/n$ (i.e. the lower the better) for n function evaluations in dimension d, Comparison between standard CMA with random Gaussian numbers, and CMA with Sobol Gaussian numbers for very small numbers of iterations in dimension 4 and 16. Sobol points are equivalent to random points, whereas Halton points (without scrambling) lead to very poor results. Results in bold face are results in which a statistical difference with the random case appeared in the good direction (better than the random case); italic font is used for results in which a statistical difference occured in favor of usual random points: the difference is most often in favor of random points for Halton; and always in favor of Sobol except for the "fschwefelmult" function.

Problem	CMA	DCMA (Halton)	DCMA (Sobol)
10 function-evaluations in dimension 4			
fsphere	-0.169	*0.00688*	**-0.209**
fcigar	5.12	5.06	**4.97**
fstepsphere	-2.2	-1.58	**-2.84**
fconcentric	0.126	*0.15*	**0.0793**
fgriewank	-0.69	*-0.565*	**-0.952**
frastrigin	1.25	*1.29*	1.25
fschwefelmult	2.96	**2.96**	*2.96*
fsectorsphere	2.25	*4.05*	**1.95**
40 function-evaluations in dimension 16			
fsphere	0.648±0.0499	0.677±0.169	**0.547±0.0580**
fcigar	6.18±0.020	6.2±0.182	**6.14±0.0197**
fstepsphere	0.865±0.0330	0.79±0.146	**0.733±0.0585**
fconcentric	0.32±0.0108	*0.325±0.0405*	**0.297±0.0105**
fgriewank	-0.253±0.0365	-0.238±0.161	**-0.351±0.0353**
frastrigin	1.96±0.00719	1.98±0.0538	**1.94±0.00641**
fschwefelmult	**3.52±8.38e-05**	*3.52±0.000283*	*3.52±6.72e-05*
fsectorsphere	4.9±0.0912	4.96±1.17	4.93±0.0895

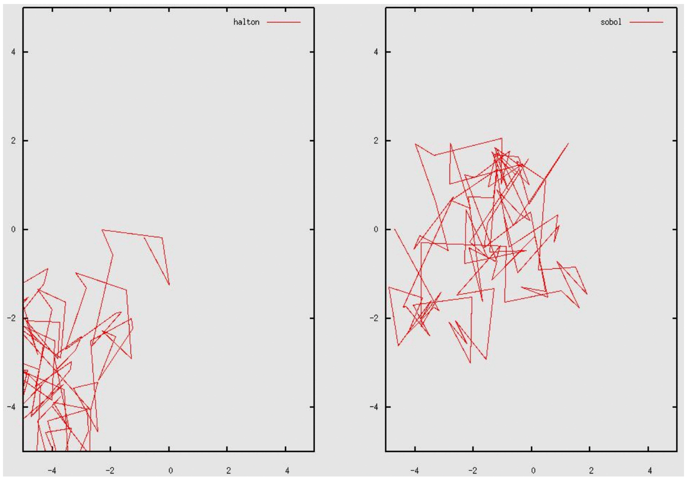


Fig. 1. 100 points of typical quasi-random walks with Gaussian quasi-random numbers generated with Halton (left) and Sobol (right). In the case of Halton, a strong short-term bias appears: the quasi-random walk goes away to the south-west.

4th variables of a Halton sequence. As well known for Halton's sequence (unscrambled version), we have long short term bias, leading to a quasi-random walk going away from 0. Figure 1 (right) shows also a quasi-random walk, generated with the partial sum of quasi-random Gaussian numbers generated with Sobol's sequence. The figure is very similar, visually, to what happens with a random walk.

It has already been pointed out in [22] that randomly rotating quasi-random Gaussian points at each offspring, in order to avoid some presumed bias, in useless and in fact reduces the efficiency of quasi-random points. We therefore here only use the standard version proposed above, without adding such rotation.

3 Quasi-random Mutations Improve the Probability of Finding a Needle

It has been suggested that quasi-random points might be suitable only in regular cases, as they might be just more able of benefiting from artificial symetries in the problem. In order to test this assumption, we now consider the needle problem, defined as follows:

- The fitness value of any point at distance $\leqslant 1$ of $(1, \ldots, 1)$ is 0.
- The fitness value of any other point is $1+R$, where R is a random independent uniform noise on $[0, 1]$.

The needle is found if at least one point of null fitness is found. We also defined the difficult needle problem, in which:

Table 2. Results of quasi-random points on the needle functions: probability of finding the needle (the higher the better). Quasi-random points work better.

Dimension	Number of fitness-evaluations	Probability of finding (CMA)	Probability of finding (DCMA)
3	64	36%± 3%	49 % ±3%
4	64	9 %± 2%	14 % ±2%
5	64	2.6 %± 1.3 %	7.3 % ± 2.1%

Standard needle

Dimension and K	Number of fitness-evaluations	Probability of finding (CMA)	Probability of finding (DCMA)
3, $K = 3$	64	0.8 % ± 0.4 %	2.4 % ± 0.6 %
4, $K = 2$	64	9.6 % ± 1.3 %	12.0 % ± 1.5%

Difficult needle

– The fitness value of any point at distance $\leqslant 1$ of $(K, K, \ldots, K)$ is 0.
– the fitness value of any other point is $1+R$, where R is a random independent uniform noise on $[0, 1]$.

As previously, the needle is found if at least one point of null fitness is found. The results are presented in table 2.

4 Quasi-random Mutations Improve the Convergence Rate

[22] has already strongly pointed out this fact, therefore we only briefly confirm these results in Figure 2. Except for "fschwefelmult", significant (95% confidence) results in favor of DCMA occur for all fitness functions for 2560 function-evaluations. We also present in table 3 the average log of fitness values for 10240 fitness evaluations in dimension 4 to see the asymptotic behavior for some fitness functions. Quasi-random points are better in all significant comparisons.

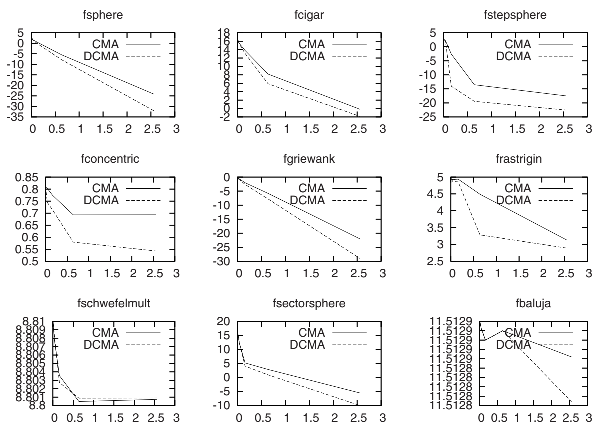


Fig. 2. Convergence rate of CMA and DCMA with Sobol in dimension 16. $Log(fitness)$ vs thousands of function evaluations. These objective functions are to be minimized. Standard deviations are not shown for the sake of readability; see text for significance.

Table 3. Comparison of standard random numbers and quasi-random numbers from the point of view of the convergence rate. The numbers are the average log of fitness values (median between (.)); the quasi-random version is almost always significantly better (the lower the better).

Problem	CMA	DCMA
fsphere	-0.0140±0.000123 (-0.013)	**-0.0147±0.00014** (-0.0147)
fcigar	-0.0139±0.000150 (-0.0139)	**-0.0150±0.000125** (-0.0149)
fstepsphere	-0.00491±0.00128 (-0.00494)	**-0.00927±0.00061** (-0.00989)
fconcentric	-0.00088±3.99e-05 (-0.0009)	**-0.00109±3.65e-05** (-0.00113)
fgriewank	-0.0139±0.000175 (-0.0139)	**-0.0167±0.000174** (-0.0174)
frastrigin	8.41e-05±0.00018 (0.000427)	-0.000144±0.000222 (0.000427)
fschwefelmult	0.00289±1.20e-07 (0.00289)	0.00289±2.37e-08 (0.00289)
fsectorsphere	-0.0127±0.00012 (-0.0128)	**-0.0136±0.000147** (-0.0136)
fbaluja	-0.00187±3.88e-05 (-0.00186)	**-0.0019±3.15e-05** (-0.00194)

5 Quasi-random Mutations Improve the Non-asymptotic Behavior: No Log

Many papers consider the logarithm of the distance to the optimum, normalized by the dimension and/or the number of iterations, as the main criterion of quality of an optimization algorithm. The advantage of this approach is that the asymptotic behavior of continuous optimization algorithms, which is usually linear for evolution strategies, is clearly visible on such plots. However, the drawback of this approach is that the focus is on the convergence rate, and not on the probability of finding a good optimum.

In order to clearly point out the weakness of this criterion for multimodal optimization, let's consider the use of this criterion for evaluating a standard algorithm with known poor results for fitness functions with local minima.

Consider simply Newton's method with random initial point. The log of the distance ε_n (after n function evaluations) to the optimum is, for this algorithm, exponential as a function of n, if the initial point is sufficiently good. Consider p the probability of an initial point ensuring that the log-precision is at most k^{-2^n}, for some fixed $k > 1$. Then, the criterion $-\log(\varepsilon_n)/n$ has expectation at least $p\log(\varepsilon_n)/n = -p2^n \log(k)/n$. This criterion therefore tends to infinity for this Newton algorithm as $n \to \infty$, whenever the function is strongly non-convex and p is close to 0! We have therefore shown that this criterion will prefer an algorithm which converges with possibly very small probability, provided that n is sufficiently large, in front of any algorithm with linear convergence "only".

On the other hand, criteria like the expected fitness value certainly not have this behavior and show much more efficiently the probability of finding the optimum, in particular in the non-asymptotic behavior. We therefore present below the average fitness value for fixed dimensions and numbers of fitness-evaluations. We point out that the same tables with the logarithm also lead to significant

Table 4. Average fitness value after 40 function evaluations (the lower, the better)

Problem	CMA	DCMA
	Dimension 2	
fsphere	0.0898±0.0356 (0.0172)	**0.0229±0.00762** (0.00436)
fcigar	21962.2±9678.1 (829.142)	**2415.2±726.955** (192.588)
fstepsphere	0.0660±0.0170 (3.36e-12)	0.0294±0.0142 (2.44e-12)
fconcentric	0.28±0.0501 (0.242)	**0.174±0.0432** (0.0972)
fgriewank	0.0282±0.0116 (0.00839)	**0.00620±0.00221** (0.0009)
frastrigin	2.94±0.237 (1.90)	**2.30±0.18** (1.51)
fschwefelmult	129.769±7.87 (100.287)	**108.386±7.04** (81.0)
fsectorsphere	2564.66±1540.54 (0.180)	0.491±0.0555 (0.175)
fbaluja	19661.4±2082.63 (18688.9)	**13146.7±1993.09** (8702.22)
	Dimension 4	
fsphere	0.399±0.0412 (0.191)	**0.285±0.0425** (0.115)
fcigar	191315±34006.3 (83925.1)	**120477±18146.6** (49928.2)
fstepsphere	0.543±0.129 (0.311)	**0.188±0.0770** (0.0252)
fconcentric	0.243±0.0159 (0.177)	0.237±0.0143 (0.187)
fgriewank	0.069±0.0103 (0.0359)	**0.0408±0.00580** (0.0202)
frastrigin	5.25±0.413 (3.94)	4.65±0.292 (3.28)
fschwefelmult	240.185±28.52 (196.578)	**149.024±19.9** (134.868)
fsectorsphere	44113.8±24239.1 (2.25)	28279.3±12841.1 (1.23)
fbaluja	11035.8±652.625 (8418)	11828.5±741.584 (9322.88)
	Dimension 16	
fsphere	1.93±0.138 (1.44)	1.7±0.120 (1.186)
fcigar	1.99e+06±143401 (1.39e+06)	1.72e+06±131583 (1.14e+06)
fstepsphere	1.93±0.133 (1.28)	1.75±0.112 (1.2)
fconcentric	0.310±0.0483 (0.199)	**0.199±0.0400** (0.134)
fgriewank	0.0689±0.00469 (0.0486)	0.0769±0.00493 (0.0570)
frastrigin	14.6±0.831 (11.5)	15.2±0.915 (11.6)
fschwefelmult	501.41±34.8 (351.154)	489.47±33.4 (324.699)
fsectorsphere	673136±57820.7 (414916)	595032±52901.3 (341269)
fbaluja	7774.31±507.891 (5136.37)	7646.29±483.262 (5719.69)

results in favor of DCMA - however, the results are more impressive with the expected fitness values as presented below.

The results in table 4 are the non-asymptotic counterpart of results in section 4 (which was convergence rate analysis). We here use 40 fitness-evaluations, for various dimensionality. [14] has already pointed out the non-asymptotic effect of quasi-random points in the simpler case of quasi-random search.

6 Quasi-random Mutations Deal Efficiently with Non-convex Functions

Non-convex functions (multimodal functions, but also monomodal functions with plateaus and other non-convex functions like "fbaluja") are typically important

Table 5. Efficiency of quasi-random points for non-convex functions from a non-asymptotic point of view. The numbers are the average fitness values (the lower the better). For fschwefelmult, the significance holds but is hidden in late digits.

Problem	CMA	DCMA
10 function evaluations		
fstepsphere	1.5±0.146 (1)	**1.06±0.10** (1)
fconcentric	1.32±0.0261 (1.33)	**1.25±0.0259** (1.2)
fgriewank	0.209±0.0287 (0.143)	**0.138±0.0225** (0.100)
frastrigin	24.6±0.642 (24.5)	24.3±0.541 (24.1)
fschwefelmult	**1670.59±0.684** (1670.75)	1671.69±0.445 (1671.45)
fbaluja	99999.4±0.0267 (99999.4)	**99999.3±0.0348** (99999.4)
40 function evaluations		
Problem	CMA	DCMA
fstepsphere	1.5±0.32 (1)	**0.5±0.18** (1e-11)
fconcentric	1.41±0.110 (1.37)	**1.05 64±0.0981** (0.972)
fgriewank	0.188±0.0588 (0.0911)	**0.054±0.0138** (0.0308)
frastrigin	26.9±1.00 (27.0)	**23.9±0.876** (23.7)
fschwefelmult	1664.87±0.703 (1664.81)	**1662.15±0.378** (1661.48)
fbaluja	99999.2±0.108 (99999.3)	**99998.8±0.2** (99999.2)
160 function evaluations		
fstepsphere	0.4±0.128 (1e-11)	**0.125±0.0853** (1e-11)
fconcentric	0.7±0.096 (0.587)	**0.51±0.0623** (0.400)
fgriewank	0.0042±0.0015 (0.00237)	**0.000659±0.000481** (0.000153)
frastrigin	20.604±2.49 (21.5)	**6.83±1.74** (4.16)
fschwefelmult	1660.21±0.014 (1660.22)	**1660.16±0.00206** (1660.16)
fbaluja	99995.6±0.938 (99996.8)	**99986.3±2.86** (99991.8)
640 function evaluations		
fstepsphere	0.388±0.143 (1e-11)	**0.0555±0.0555** (1e-11)
fconcentric	0.232±0.0394 (0.128)	**0.120±0.0384** (0.0549)
fgriewank	0.000833±0.000411 (5.42e-11)	0.000245±0.000112 (1.51e-14)
frastrigin	5.57±0.427 (3.70)	**3.64±0.204** (2.98)
fschwefelmult	1660.15±5.98e-09 (1660.15)	**1660.15±6.90e-13** (1660.15)
fbaluja	94926.9±1036.12 (95873.5)	**13597.7±2714.8** (10272.5)

fitness functions for which the convergence rate is moderately interesting: diversity loss in a plateau (or in an almost plateau) or premature convergence in a local minimum are a strong trouble. We here investigate the question of a possible loss of efficiency of quasi-random mutations for non-convex functions as a counterpart of positive effects pointed out in other sections of this paper. Table 5 show the average fitness value (no log) of the best individual of the last offspring for various non-convex fitness functions. Very strong improvements sometimes appear with quasi-randomisation, and results were almost always significantly improved. All results below hold in dimension 4.

Table 6. Mean log(*fitness*) and mean *fitness* in the case of noise (see text for details). The lower, the better. Quasi-random numbers are almost always better.

Problem	CMA	DCMA
fsphere	-0.12±0.00788	**-0.169±0.00469**
fcigar	0.013±0.0022	**-0.0205±0.00196**
fstepsphere	-0.341±0.00328	-0.344±0.00246
fconcentric	-0.0252±0.00096	**-0.030±0.00109**
fgriewank	-0.117±0.00490	**-0.167±0.0048**
frastrigin	-0.00444±0.00283	**-0.015±0.00413**
fschwefelmult	**0.053±0.00152**	0.0580±0.0014
fsectorsphere	-0.0391±0.0062	**-0.0829±0.00481**
fbaluja	**0.111±0.00142**	0.11±0.00141

log(*fitness*)

Problem	CMA	DCMA
fsphere	0.000768±0.000372	**5.78e-06±2.55e-06**
fcigar	2819.93±2594.19	0.823±0.120
fstepsphere	0.0018±0.00138	0.00432±0.00422
fconcentric	0.207±0.0136	**0.163±0.0146**
fgriewank	0.00418±0.001	**0.000492±0.000297**
frastrigin	2.00±0.300	**1.20±0.224**
fschwefelmult	**111.731±10.198**	152.172±13.092
fsectorsphere	0.165±0.0641	**0.00304±0.000894**
fbaluja	**12775.2±1035.92**	17218±1348.25

fitness value

7 Quasi-random Mutations Deal Efficiently with Noise

Finally we tested fitness functions corrupted by noise. We just replaced the fitness function by its product with a random independent uniform number in $[0, 1]$, and we get the table 6 of results in dimension 2 with 160 function-evaluations.

In fact, the results are *more* impressive than the results of the non-noisy case: there's no decay of performance in the noisy case. We tested both the log and the no-log cases.

8 Conclusion

We have in this paper shown how general is the improvement induced by quasi-random numbers. In particular, quasi-random mutations lead to better results, not only from the point of view of the convergence rate , but also for several notions of robustness:

– The improvement for the convergence rate scales from a few percents in the case of the sphere function or the cigar function to 80 % of speed-up in the case of the step-sphere function in dimension 4 - this suggests that is is much more the behavior in front of plateaus or needles which is improved, and not the behavior in a regular, smooth framework;
– The improvement remains when the fitness function is corrupted by noise; in fact, all comparisons are seemingly much more impressive when the fitness is corrupted by noise! Therefore, we claim that the effect ot quasi-random numbers are not limited to artificial smooth cases.
– The improvement is much more impressive on expected fitness values than on the convergence rate, because the quasi-random points improve the robustness and the probability of finding a solution as well as (and more than) the asymptotic convergence rate;

- In the case of small numbers of fitness values, one can also trust quasi-random points; the non-asymptotic behavior for non-logarithmic views on the fitness function (section 5) and small dimension provides impressive results: average fitness values are divided by factors between 3.9 (sphere function) and 9.1 (cigar function) in dimension 2. The results are less impressive in higher dimension (factor 1.16 in dimension 16 for the cigar function).
- Quasi-random points are not afraid of non-convex fitness functions; we see 51% of improvement for the Griewank function with 10 function-evaluations in table 5, 3.2 for the step-sphere function and 53% for the Rastrigin function with 160 function-evaluations. In some other cases, the improvement is negligible, but we point out that it is never a loss of efficiency.
- The probability of finding a needle is improved also; the more difficult the needle problem, the higher the improvement; we conjecture that the improvement would be much higher with higher values of λ.

These results suggest that the improvement is due to (i) a better distribution of mutations inside one offspring but also in a cumulative manner over multiple steps as shown by earlier results in [22] (i.e., increasing independence between offsprings by random rotations of each offspring reduces the efficiency) (ii) less importantly, a better convergence rate by a better estimate of the position of the optimum - but this effect is probably much bigger for higher values of λ. We also tested the rate at which the covariance matrix is evaluated, but this effect is seemingly very small - perhaps this could be improved by better update rules. (i) includes/explains (a) the "init" effect is we consider as initialization the initial offsprings (b) the good "needle" effect (the step-sphere function, on which results are quite good, is typically a "multiple" needle problem, whereas (ii) explains the better convergence rate on the sphere function.

We also point out that replacing Monte-Carlo mutations by Quasi-Monte-Carlo points is straightforward: if your Gaussian points are generated thanks to the use of the reverse probability distribution function of the Gaussian on random uniform points, then just replace random independent vectors uniform on $[0, 1]$ by some quasi-random *good* generator. In particular, Sobol's sequence is very efficient and as fast as random points.

A further work is the design of step-size adaptation rules adapted to Quasi-Monte-Carlo mutations: the usual derivation of the cumulative step-size adaptation is not adapted with quasi-random mutations. However, results in this paper are quite stable and convincing without such adaptation; in all this paper we have used a standard cumulative step-size adaptation.

Acknowledgements. The authors thank the many people who provided feedback on previous results around quasi-random for providing several of the questions which are answered in this article. We also thank A. Auger and N. Hansen for interesting feedback, in particular with respect to the future derivation of step-size adaptation rules for quasi-randomized mutations.

References

1. Atanassov, E.I.: On the discrepancy of the halton sequences. Math. Balkanica 18(12), 15–32 (2004)
2. Braaten, E., Weller, G.: An improved low-discrepancy sequence for multidimensional quasi-monte carlo integration. J. Comput. Phys. 33, 249–258 (1979)
3. Bratley, P., Fox, B.: Algorithm 659: Implementing sobol's quasirandom sequence generator. ACM Transactions on Mathematical Software 14(1), 88–100 (1988)
4. Cervellera, C., Muselli, M.: A deterministic learning approach based on discrepancy. In: Apolloni, B., Marinaro, M., Tagliaferri, R. (eds.) WIRN 2003. LNCS, vol. 2859, pp. 53–60. Springer, Heidelberg (2003)
5. Cranley, R., Patterson, T.N.L.: Randomization of number theoretic methods for multiple integration. SIAM J. Numer. Anal. 13(6), 904–914 (1976)
6. Faure, H.: Good permutations for extreme discrepancy. J. Number Theory 42, 47–56 (1992)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 11(1) (2003)
8. Hardy, G.H.: On double fourier series, and especially those which represent the double zeta-function with real and incommensurable parameters. Quart. J. Mathematics 37, 53–79 (1905)
9. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. Math. Comp. 67, 299–322 (1998)
10. Kimura, S., Matsumura, K.: Genetic algorithms using low-discrepancy sequences. In: GECCO, pp. 1341–1346 (2005)
11. L'Ecuyer, P., Lemieux, C.: Recent Advances in Randomized Quasi-Monte Carlo Methods, pp. 419–474. Kluwer Academic Publishers, Dordrecht (2002)
12. Mascagni, M., Chi, H.: On the scrambled halton sequence. Monte Carlo Methods Appl. 10(3), 435–442 (2004)
13. Morokoff, W.J., Caflish, R.E.: Quasi-random sequences and their discrepancies. SIAM J. Sci. Comput. 15(6), 1251–1279 (1994)
14. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods (1992)
15. Okten, G., Srinivasan, A.: Parallel quasi-monte carlo methods on a heterogeneous cluster. In: Niederreiter, H., Fang, K.-T., Hickernell, F.J. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 406–421. Springer, Heidelberg (2002)
16. Owen, A.B.: Quasi-Monte Carlo Sampling, A Chapter on QMC for a SIGGRAPH 2003 course (2003)
17. Sarkar, P.K., Prasad, M.A.: A comparative study of pseudo and quasi random sequences for the solution of integral equations. J. Computational Physics 68, 66–88 (1978)
18. Sloan, I.H., Woźniakowski, H.: When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? Journal of Complexity 14(1), 1–33 (1998)
19. Sobol, I.M.: On the systematic search in a hypercube. Siam journal on Numerical Analysis 16(5), 790–793 (1979)
20. Srinivasan, A.: Parallel and distributed computing issues in pricing financial derivatives through quasi-monte carlo. In: Proceedings of the 16th International Parallel and Distributed Processing Symposium (2002)
21. Teytaud, O., Gelly, S., Mary, J.: Active learning in regression, with application to stochastic dynamic programming. In: Proceedings of ICINCO 2007, pp. 47–54 (2007)

22. Teytaud, O., Gelly, S.: Dcma: yet another derandomization in covariance-matrix-adaptation. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 955–963. ACM Press, New York (2007)
23. Tuffin, B.: On the use of low discrepancy sequences in monte carlo methods (1996)
24. Tuffin, B.: A new permutation choice in halton sequences. Monte Carlo and Quasi-Monte Carlo 127, 427–435 (1997)
25. van der Corput, J.G.: Verteilungsfunktionen. Proc. Ned. Akad. v. Wet. 38, 813–821 (1935)
26. Vandewoestyne, B., Cools, R.: Good permutations for deterministic scrambled halton sequences in terms of l2-discrepancy. Computational and Applied Mathematics 189(1,2), 341–361 (2006)
27. Wang, X., Hickernell, F.: Randomized halton sequences. Math. Comput. Modelling 32, 887–899 (2000)
28. Warnock, T.T.: Computational investigations of low-discrepancy point sets. In: Zaremba, S.K. (ed.) Applications of Number Theory to Numerical Analysis (Proceedings of the Symposium). University of Montreal, pp. 319–343 (1972)
29. Warnock, T.T.: Computational investigations of low-discrepancy point sets ii. In: Niederreiter, H., Shiue, P.J.-S. (eds.) Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing. Springer, Berlin (1995)

Fitness Expectation Maximization

Daan Wierstra[1], Tom Schaul[1], Jan Peters[3], and Jürgen Schmidhuber[1,2]

[1] IDSIA, Galleria 2, 6298 Manno-Lugano, Switzerland
{daan,tom,juergen}@idsia.ch
[2] TU Munich, Boltzmannstr. 3, 85748 Garching, München, Germany
[3] Max Planck Institute for Biological Cybernetics, Tübingen, Germany
mail@jan-peters.net

Abstract. We present Fitness Expectation Maximization (FEM), a novel method for performing 'black box' function optimization. FEM searches the fitness landscape of an objective function using an instantiation of the well-known Expectation Maximization algorithm, producing search points to match the sample distribution weighted according to higher expected fitness. FEM updates both candidate solution parameters and the search policy, which is represented as a multinormal distribution. Inheriting EM's stability and strong guarantees, the method is both elegant and competitive with some of the best heuristic search methods in the field, and performs well on a number of unimodal and multimodal benchmark tasks. To illustrate the potential practical applications of the approach, we also show experiments on finding the parameters for a controller of the challenging non-Markovian double pole balancing task.

1 Introduction

Real-valued 'black box' function optimization is one of the major topics in modern applied machine learning research (e.g. see [1]). It concerns itself with optimizing the continuous parameters of an unknown (black box) objective fitness function, the exact analytical structure of which is assumed to be unknown or unspecified. Specific function measurements can be performed, however. The goal is to find a reasonably high-fitness candidate solution while keeping the number of function measurements limited. The black box optimization framework is crucial for many real-world domains, since often the precise structure of a problem is either not available to the engineer, or too expensive to model or simulate.

Now, since exhaustively searching the entire space of solution parameters is considered to be infeasible, and since we do not assume we have access to a precise model of our fitness function, we are forced to settle for trying to find a reasonably good solution that satisfies certain pre-specified constraints. This, inevitably, involves using a sufficiently intelligent heuristic approach, since in practice it is important to find the right domain-specific trade-off on issues such as convergence speed, expected quality of the solutions found and the algorithm's sensitivity to local suboptima on the fitness landscape.

A variety of algorithms has been developed within this framework, including methods such as Simulated Annealing [2], Simultaneous Perturbation Stochastic Approximation [3], the Cross-Entropy method [4,5], and evolutionary methods such as Covariance Matrix Adaption (CMA) [6] and the class of Estimation of Distribution Algorithms (EDAs) [7].

In this paper, we postulate the similarity and actual equivalence of black box function optimization and one-step reinforcement learning. In our attempt to create a viable optimization technique based on reinforcement learning, we fall back onto a classical goal of reinforcement learning (RL), i.e., we search for a way to reduce the reinforcement learning problem to a supervised learning problem. In order to do so, we re-evaluate the recent result in machine learning, that reinforcement learning can be reduced onto *reward-weighted regression* [8] which is a novel algorithm derived from Dayan & Hinton's [9] expectation maximization (EM) perspective on RL. We show that this approach generalizes from reinforcement learning to fitness maximization to form Fitness Expectation Maximization (FEM), a relatively well-founded instantiation of EDAs which relates to other (EM-inspired) methods for optimization (e.g. see [10,11]).

This algorithm is tested on a set of unimodal and multimodal benchmark functions, and is shown to exhibit excellent performance on both unimodal and multimodal benchmarks. A defining feature of FEM is its adaptive *search policy*, which takes the form of a multinormal distribution that produces *correlated* search points in search space. Its covariance matrix makes the algorithm invariant across rotations in the search space, and enables the algorithm to fine-tune its search appropriately, resulting in arbitrarily high-precision solutions. Furthermore, using the stability properties of the EM algorithm, the algorithm seeks to avoid catastrophically greedy updates on the search policy, thus preventing premature convergence in some cases.

The paper is organized as follows. The next section provides a quick overview of the general problem framework of real-valued black box function optimization. The ensuing sections describe the derivation of the EM-based algorithm, the concept of 'fitness shaping', and the online instantiation of our algorithm. The experiments section shows initial results with a number of unimodal and multimodal benchmark problems. Furthermore, results with the non-Markovian double pole balancing problem are discussed. The paper concludes with a discussion on the advantages and problems of the method, and points to some possible directions for future extensions.

2 Algorithm Framework

First let us introduce the algorithm framework and the corresponding notation. The objective is to optimize the n-dimensional continuous vector of objective parameters $\mathbf{x}$ for an unknown fitness function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The function is unknown or 'black box', in that the only information accessible to the algorithm consists of function measurements selected by the algorithm. The goal is to optimize $f(\mathbf{x})$, while keeping the number of function evaluations – which are considered costly – as low as possible. This is done by successively evaluating batches of a number $1 \dots N$ of separate search points $\mathbf{z}_1 \dots \mathbf{z}_N$ on the fitness function, while using the information extracted from fitness evaluations $f(\mathbf{z}_1) \dots f(\mathbf{z}_N)$ to adjust both the current candidate solution $\mathbf{x}$ and the search policy defined as a Gaussian with mean $\mathbf{x}$ and covariance matrix $\mathbf{\Sigma}$.

3 Expectation Maximization for Black Box Function Optimization

At every point in time while running the algorithm, we want to optimize the expected fitness $J = \mathbf{E}_{\mathbf{z}}[f(\mathbf{z})]$ of the next batch, given the current batch of search samples. We assume that every batch g is generated by search policy $\pi^{(g)}$ parameterized by $\theta = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$, representing the current candidate solution $\mathbf{x}$ and covariance matrix $\mathbf{\Sigma}$.

In order to adjust parameters $\theta = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$ towards solutions with higher associated fitness, we *match* the search distribution to the actual sample points, but weighted by their utilities. Now let $f(\mathbf{z})$ be the fitness at a particular search point $\mathbf{z}$, and, utilizing the familiar multivariate normal distribution, let $\pi(\mathbf{z}|\theta) = \mathcal{N}(\mathbf{z}|\mathbf{x}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{z}-\mathbf{x})^{\mathbf{T}}\mathbf{\Sigma}^{-1}(\mathbf{z}-\mathbf{x})\right]$ denote the probability density of search point $\mathbf{z}$ given the current search policy π. The expectation

$$J = \mathbf{E}_{\mathbf{z}}[f(\mathbf{z})] = \int \pi(\mathbf{z}|\theta) f(\mathbf{z}) d\mathbf{z}.$$

indicates the expected fitness over all possible sample points, weighted by their respective probabilities under policy π.

3.1 Optimizing Utility-Transformed Fitness

While an objective function such as the above is sufficient in theory, algorithms which simply optimize it have major disadvantages. They might be too aggressive when little experience – few sample points – is available, and converge prematurely to the best solution they have seen so far. On the opposite extreme, they might prove to be too passive and be biased by less fortunate experiences. Trading off such problems has been a long-standing challenge in reinforcement learning. However, in decision theory, such problems are surprisingly well-understood [12]. In that framework it is common to introduce a so-called utility transformation $u(f(z))$ which has to fulfill the requirement that it scales monotonically with f, is semi-positive and integrates to a constant. Once a utility transformation is inserted, we obtain an expected utility function given by

$$J_u(\theta) = \int p(\mathbf{z}|\theta) u(f(\mathbf{z})) d\mathbf{z}. \quad (1)$$

The utility function $u(f)$ is an adjustment for the aggressiveness of the decision making algorithms, e.g., if it is concave, it's attitude is risk-averse while if it is convex, it will be more likely to consider a fitness more than a coincidence. Obviously, it is of essential importance that this risk function is properly set in accordance with the expected fitness landscape, and should be regarded as a metaparameter of the algorithm. Notice the similarity to the selection operator in evolutionary methods.

We have empirically found that rank-based shaping functions (rank-based selection) work best for various problems, also because they circumvent the problem of extreme fitness values disproportionately distorting the estimation of the search distribution, making careful adaptation of the forget factor during search unnecessary even for problems with wildly fluctuating fitness. In this paper, we will consider a simple rank-based utility transformation function, the piecewise linear $u_k = u(f(\mathbf{z}_k)|f(\mathbf{z}_{k-1}), \ldots, f(\mathbf{z}_{k-N}))$ which first ranks all samples $k-N, \ldots, k$ based on fitness value, then assigns zero to the $N-m$ worst ones and assigns values linearly from $0 \ldots 1$ to the m best samples.

3.2 Fitness Expectation Maximization

Analogously as in [8,9], we can establish the lower bound

$$\begin{aligned} \log J_u(\theta) &= \log \int q(\mathbf{z}) \frac{p(\mathbf{z}|\theta)u(f(\mathbf{z}))}{q(\mathbf{z})} d\mathbf{z} && (2)\\ &\geq \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\theta)u(f(\mathbf{z}))}{q(\mathbf{z})} d\mathbf{z} && (3)\\ &= \int q(\mathbf{z}) \left[\log p(\mathbf{z}|\theta) + \log u(f(\mathbf{z})) - \log q(\mathbf{z})\right] d\mathbf{z} && (4)\\ &:= \mathcal{F}(q, \theta), && (5)\end{aligned}$$

due to Jensen's inequality with the additional constraint $0 = \int q(\mathbf{z})d\mathbf{z} - 1$. This points us to the following EM algorithm:

Proposition 1. *An Expectation Maximization algorithm for both optimizing expected utility and the raw expected fitness is given by*

$$\textit{E-Step:}\quad q_{g+1}(\mathbf{z}) = \frac{p(\mathbf{z}|\theta)u(f(\mathbf{z}))}{\int p(\tilde{\mathbf{z}}|\theta)u(f(\tilde{\mathbf{z}}))d\tilde{\mathbf{z}}}, \quad (6)$$

$$\textit{M-Step Policy:}\quad \theta_{g+1} = \arg\max_{\theta} \int q_{g+1}(\mathbf{z}) \log p(\mathbf{z}|\theta) d\mathbf{z}. \quad (7)$$

Proof. The E-Step is given by $q = \operatorname{argmax}_q \mathcal{F}(q, \theta)$ while fulfilling the constraint $0 = \int q(\mathbf{z})d\mathbf{z} - 1$. Thus, we have a Lagrangian $L(\lambda, q) = \mathcal{F}(q, \theta) - \lambda$. When differentiating $L(\lambda, q)$ with respect to q and setting the derivative to zero, we obtain $q^*(\mathbf{z}) = p(\mathbf{z}|\theta)u(f(\mathbf{z}))\exp(\lambda - 1)$. We insert this back into the Lagrangian obtaining the dual function $L(\lambda, q^*) = \int q^*(\mathbf{z})d\mathbf{z} - \lambda$. Thus, by setting $dL(\lambda, q^*)/d\lambda = 0$, we obtain $\lambda = 1 - \log \int p(\mathbf{z}|\theta)u(f(\mathbf{z}))d\mathbf{z}$, and solving for q^* implies Eq (6). The M-steps compute $\theta_{g+1} = \operatorname{argmax}_\theta \mathcal{F}(q_{g+1}, \theta)$.

In practice, when using a Gaussian search distribution parameterized by $\theta^{(k)} = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$, the EM process comes down to simply fitting the samples in every batch to the Gaussian, weighted by the utilities.

4 Online Fitness Expectation Maximization

In order to speed up convergence, the algorithm can be executed *online*, that is, sample by sample, instead of batch by batch. The online version of the algorithm can yield superior performance since updates to the policy can be made at every sample instead of just once per batch, and because doing so tends to preserve sample diversity better than by using the batch version of the algorithm. Crucial is that a *forget factor* α is now introduced to modulate the speed at which the search policy adapts to the current sample. Batch size N is now only used for utility ranking function u which ranks the current sample among the N last seen samples. The resulting FEM algorithm pseudocode can be found in Algorithm 1.

Algorithm 1. Fitness Expectation Maximization

```
use shaping function $u$, batch size $N$, forget factor $\alpha$
$k \leftarrow 1$
initialize search parameters $\theta^{(k)} = \langle \mathbf{x}, \mathbf{\Sigma} \rangle$
repeat
    draw sample $\mathbf{z}_k \sim \pi(\mathbf{x}, \mathbf{\Sigma})$
    evaluate fitness $f(\mathbf{z}_k)$
    compute rank-based fitness shaping $u_k = u(f(\mathbf{z}_k) | f(\mathbf{z}_{k-1}), \ldots, f(\mathbf{z}_{k-N}))$
    $\mathbf{x} \leftarrow (1 - \alpha u_k)\mathbf{x} + \alpha u_k \mathbf{x}$
    $\mathbf{\Sigma} \leftarrow (1 - \alpha u_k)\mathbf{\Sigma} + \alpha u_k (\mathbf{x} - \mathbf{z}_k)(\mathbf{x} - \mathbf{z}_k)^{\mathbf{T}}$
    $k \leftarrow k + 1$
until stopping criterion is met
```

5 Experiments

5.1 Standard Benchmark Functions

Good test functions should be easy to interpret, but scale up with n. They must be highly nonlinear, non-separable, largely resistant to hill-climbing, and preferably contain deceptive local suboptima. To test the performance of the algorithm, we chose 6 unimodal functions (Sphere, Schwefel, Tablet, Cigar, Different-Powers, Ellipsoid) and 4 multimodal functions (Ackley, Rastrigin, Weierstrass and Griewank) from a set of benchmark functions from [13] and [6] that are typically used in the literature, for comparison purposes and for competitions. As those functions are designed to be minimized, we take the fitness to be the negative function value. The multimodal functions were tested with both FEM and the Covariance Matrix Adaptation (CMA) [6] algorithm – widely regarded as one of the premier algorithms in this field – for comparison purposes.

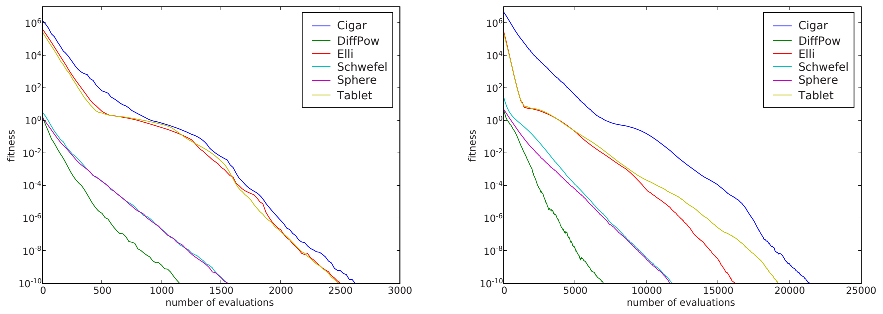


Fig. 1. Results for experiments on the unimodal benchmark functions. Left: dimensionality 5, right: dimensionality 15.

In order to prevent potentially biased results, and to avoid trivial optima (e.g. at the origin), we follow [13] and consistently transform (by a combined rotation and translation) the functions' inputs in order to make the variables non-separable. This immediately renders many direct search method virtually useless, since they cannot cope with correlated search directions, unlike FEM and CMA.

The tunable parameters of the FEM algorithm are comprised of batch size N, the fitness shaping function u applied on the fitness function f and forget factor α. The parameters should be chosen by the expert to fit the expected ruggedness of the fitness landscape. The forget factor must be low enough such that it does not too quickly forget earlier successful search points. The shaping function must be chosen such that enough randomness is preserved in the search policy after every update, which entails including the lesser samples in utility attribution. For all experiments, comprising both the benchmark unimodal/multimodal functions and the non-Markovian double pole balancing task, initial $\mathbf{\Sigma}$ was set to the identity matrix $\mathbf{\Sigma} = \mathbf{I}$ and $\mathbf{x}$ was always randomly initialized as $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

We ran FEM on the set of unimodal benchmark functions with dimensions 5 and 15 using a target precision of 10^{-10}. Figure 1 shows the average number of evaluations until success over 20 runs on the unimodal functions. The parameter settings for dimensionality 5 were identical in all runs: $\alpha = 0.1$ and $N = 50$, parameter m for selecting the shaping function's top m samples was set at $m = 5$. The parameter settings for all runs in dimensionality 15 were: $\alpha = 0.02$, $N = 25$ and $m = 10$. All runs converged. The number of evaluations was roughly equal to that of CMA on the small dimensionality, and for most problems not more than a factor 3 slower, even with dimensionality 15 [6].

On the multimodal benchmark functions we performed experiments while varying the distance of the initial guess to the optimum between 1 and 100. As with the unimodal functions, the problems were appropriately translated and rotated, while the initial $\mathbf{x}$ was randomly initialized on the surface of the hypersphere with radius 1, 10 or 100 and the optimum at its center. Those runs were performed on dimension 2 with a target precision of 0.01, since here the focus

Table 1. Results for the multimodal benchmark functions. Shown are percentages of runs that found the global optimum, for both FEM and CMA, for varying starting distances.

	FEM			**CMA**		
Distance	1	10	100	1	10	100
Rastrigin	91%	87%	64%	13%	11%	14%
Ackley	100%	100%	0%	89%	70%	3%
Weierstrass	19%	9%	19%	90%	92%	92%
Griewank	100%	2%	0%	100%	2%	0%

was on avoiding local maxima. The parameter settings for the multimodal runs were: $\alpha = 0.02$, $N = 25$ and $m = 10$. Table 1 shows, for all multimodal functions, the percentage of runs where FEM found the global optimum (as opposed to it getting stuck in a local suboptimum) depending on the distance from the initial guess to the optimum. The percentages are computed over 100 runs. For comparison purposes we included the results for the CMA implementation of [6], although it must be said that in all likelihood better results can be achieved for CMA using population sizes that are larger than standard for that algorithm.

One additional, linear benchmark function $f(\mathbf{z}) = \sum_j z_j$ was tested to verify the expected premature convergence of the algorithm. Indeed, FEM converges prematurely like EDAs typically do (e.g. [14]), while CMA performed well (see e.g. [15]). This suggests the approach might not be applicable to all domains and that it might benefit from a mutative approach modeling mutations instead of weighted sample distributions.

Lastly, we performed experiments using a batch-based version of the algorithm instead of the online version. We found the standard benchmark problems could only be solved using large batch sizes (1000 and up), slowing down the algorithm considerably. This might be due to the reduced sample diversity using small batch sizes, which is ameliorated using an online update rule which only gradually adjusts $\mathbf{\Sigma}$ values.

To summarize, our experiments on these standard black box optimization benchmarks indicate that FEM is competitive with other high-performance algorithms. The premature convergence on the simple linear test function was expected and it remains to be seen whether this will affect the long-term viability of the approach. Last, the superior performance of the online version of this algorithm might indicate that the problem of diversity maintenance could prove to be an important topic of future research on FEM and EDAs in general.

5.2 Non-markovian Double Pole Balancing

Non-Markovian double pole balancing [16] can be considered a difficult benchmark task for control optimization. We use the implementation as found in [17]. The FEM algorithm optimizes the parameters of the controller, which is implemented as a simple neural network with three inputs, three hidden sigmoid neurons, and one output neuron.

Table 2. Results for non-Markovian double pole balancing. The table shows the average number of evaluations for SANE [18], ESP [17], NEAT [19], CMA [20,6], CoSyNE [21] and FEM.

Method	SANE	ESP	NEAT	CMA	CoSyNE	FEM
Evaluations	262, 700	7, 374	6, 929	3, 521	1, 249	2, 099

The algorithm's parameters were set as follows: piecewise linear shaping function with $m = 5$ (top 5 selection), forget factor $\alpha = 0.05$ and batch size $N = 50$. A run was considered a *success* when the poles did not fall over for 100, 000 time steps. The results on a total of 200 runs are, on average, 2099 evaluations until success (standard deviation: 1505). Not included in these statistics are 49/200 runs that did not reach success within the limit of 10000 evaluations, which compares badly with both CoSyNE and CMA which (almost) always converge. Table 5.2 shows results of other premier algorithms applied to this task, including CMA. All methods optimized the same type of recurrent neural network, albeit with differing numbers of hidden neurons. FEM, when it converges, outperforms all other methods except CoSyNE. Since our algorithm performs well on this relatively hard control benchmark, we expect the algorithm to do well on future real-world experiments.

6 Discussion

Fitness Expectation Maximization constitutes a simple, principled approach to real-valued black box function optimization with a rather clean derivation from first principles. Its theoretical relationship to the field of reinforcement learning and in particular reward-weighted regression should be clear to any reader familiar with both fields. We anticipate that rephrasing the black box optimization problem as a reinforcement learning problem solvable by RL methods will spawn a whole series of additional new algorithms exploiting this connection.

The experiments show that, on the unimodal and multimodal benchmarks, FEM is competitive with respect to its the main 'competitor' algorithm CMA, at least on lower dimensional problems. Taking into account the good results on the pole balancing tasks, we envision that FEM might become a serious competitor in the field of black box function optimization, especially for neuroevolution.

Future work on FEM will include a systematic study that must determine whether it can be made to outperform other search methods consistently on other typical benchmarks and real-world tasks. It remains to be seen how the method scales up with increased dimensionality, especially compared to CMA. We suggest extending the algorithm from a single multinormal distribution as search policy representation to a mixture of Gaussians (which is a common procedure for 'vanilla' EM), thus further reducing its sensitivity to local suboptima. Other pressing work includes a theoretical analysis of the shaping (selection) function, which should ideally be made to adapt automatically based on the data instead of tuned manually.

The premature convergence on the linear test function is worrisome. Future work will determine whether this phenomenon affects the practical applicability to real-world problems such as neurocontrol. Alternatively, we must investigate whether the introduction of a more mutative approach like CMA might be beneficial.

7 Conclusion

We introduced Fitness Expectation Maximization to tackle the important class of real-valued 'black box' function optimization problems. Reframing black box optimization as a one-step reinforcement learning problem, we developed a method similar in spirit to expectation maximization. Using a search policy which matches samples weighted by their utilities, the algorithm performs competitively on a standard benchmark set of unimodal and multimodal functions and non-Markovian double pole balancing control.

Acknowledgments

This research was funded by SNF grants 200021-111968/1 and 200021-113364/1.

References

1. Spall, J., Hill, S., Stark, D.: Theoretical framework for comparing several stochastic optimization approaches. Probabilistic and Randomized Methods for Design under Uncertainty, 99–117 (2006)
2. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
3. Spall, J.C.: Stochastic optimization and the simultaneous perturbation method. In: WSC 1999: Proceedings of the 31st conference on Winter simulation, pp. 101–109. ACM, New York (1999)
4. Rubinstein, R.Y., Kroese, D.P.: The Cross-Entropy Method: A Unified Approach to Monte Carlo Simulation, Randomized Optimization and Machine Learning. Springer, Heidelberg (2004)
5. De Boer, P., Kroese, D., Mannor, S., Rubinstein, R.: A tutorial on the cross-entropy method. Annals of Operations Research 134, 19–67 (2004)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
7. Larraanaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Norwell (2001)
8. Peters, J., Schaal, S.: Reinforcement learning by reward-weighted regression for operational space control. In: Proceedings of the International Conference on Machine Learning (ICML) (2007)
9. Dayan, P., Hinton, G.E.: Using expectation-maximization for reinforcement learning. Neural Computation 9(2), 271–278 (1997)

10. Wolpert, D.H., Rajnarayan, D.G.: Parametric Learning and Monte Carlo Optimization. ArXiv e-prints 704 (April 2007)
11. Gallagher, M., Frean, M., Downs, T.: Real-valued evolutionary optimization using a flexible probability density estimator. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, vol. 1, pp. 840–846. Morgan Kaufmann, San Francisco (1999)
12. Chernoff, H., Moses, L.E.: Elementary Decision Theory. Dover Publications (1987)
13. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore (2005)
14. Gonzalez, C., Lozano, J.A., Larraanaga, P.: Mathematical modelling of umdac algorithm with tournament selection. Behaviour on linear and quadratic functions. International Journal of Approximate Reasoning 31(3), 313–340 (2002)
15. Hansen, N.: An analysis of mutative σ-self-adaptation on linear fitness functions. Evolutionary Computation 14(3), 255–275 (2006)
16. Wieland, A.: Evolving neural network controllers for unstable systems. In: Proceedings of the International Joint Conference on Neural Networks, Seattle, WA, pp. 667–673. IEEE, Piscataway (1991)
17. Gomez, F.J., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior 5, 317–342 (1997)
18. Moriarty, D.E., Miikkulainen, R.: Efficient reinforcement learning through symbiotic evolution. Machine Learning 22, 11–32 (1996)
19. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10, 99–127 (2002)
20. Igel, C.: Neuroevolution for reinforcement learning using evolution strategies. In: Congress on Evolutionary Computation (CEC 2003), vol. 4, pp. 2588–2595. IEEE Press, Los Alamitos (2003)
21. Faustino Gomez, J.S., Miikkulainen, R.: Efficient non-linear control through neuroevolution. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212. Springer, Heidelberg (2006)

Formally Testing Liveness by Means of Compression Rates[⋆]

César Andrés, Ismael Rodríguez, and Fernando Rubio

Dept. Sistemas Informáticos y Computación
Facultad de Informática, Universidad Complutense
c.andres@fdi.ucm.es,{isrodrig,fernando}@sip.ucm.es

Abstract. We present a formal method to determine whether there exist *living* creatures in a given computational environment. Our proposal is based on studying the evolution of the entropy of the studied system. In particular, we check whether there exist entities decreasing the entropy in some parts, while increasing it in the rest of the world, which fits into the well-known *maximum entropy production principle*. The entropy of a computational environment is measured in terms of its *compression rate* with respect to some compression strategy. Some life-related notions such as *biodiversity* are quantified as well. These ideas are presented by means of formal definitions. A toy example where a simple living structure is identified in a video stream is presented, and some results are reported.

Keywords: Artificial Life, Maximum Entropy Principle, Compression Algorithms.

1 Introduction

Whenever the important question of *what is life* is considered, the controversy eventually arises. For elementary school students, the answer is rather simple: Live beings are those which feed themselves, relate with the environment, and reproduce. However, this definition is neither operative nor precise enough in practice. On the one hand, defining notions such as *feeding*, *relating*, and *reproducing* with enough generality to embrace all kinds of living beings existing in Nature is not easy. Moreover, if Artificial Life is considered [1,5,7,8,16], then defining these concepts is even more challenging. On the other hand, the previous definition of life ignores some living beings that do not fulfill some of the proposed conditions (e.g., *mules* do not *reproduce*).

In this regard, we may consider the Maximum Entropy Production Principle (see e.g. [2,3,9,11]). Grossly speaking, this principle states the following ideas: (a) Due to the Thermodynamics laws, the entropy of any environment must increase along time; (b) living beings are repetitive patterns that increase the order in their environment by their simple existence: Species are made of repetitive patterns (living individuals), and the parts of a living being are repetitive themselves (organs, cells, etc); so, (c) if (a) and (b) are not contradictory then living beings must generate *more* entropy around them than the entropy reduced by the existence of their bodies themselves. That is, living beings are entities with low entropy that increase the entropy around them as they live.

[⋆] Supported by projects TIN2006-15578-C02-01, PAC06-0008-6995, MRTN-CT-2003-505121.

Let us note that the definition of entropy actually depends on the kind of environment we are considering. In a chemical environment, Thermodynamics provides appropriate entropy notions. In an information environment, several notions of *order* and *entropy* are available. For instance, Shannon's Theorem [12] provides a classical definition of entropy. This notion is restricted to a *memoryless* source, i.e. the probability of each symbol is assumed to be independent of the symbols entering the remaining sites in the chain. While being appropriate for some cases, this assumption is unrealistic in most approaches. More generally, given two sequences of bits α_1 and α_2, we can find two formal criteria C_1 and C_2 to measure the information entropy such that α_1 is more *ordered* than α_2 for C_1, and it is the other way around for C_2. For instance, let us suppose that C_1 (respectively, C_2) measures the order degree of a sequence of bits in terms of the *compression rate* we achieve by applying a compression algorithm A_1 (resp. A_2) to the sequence. If the resulting compressed sequence is *short* then it means that the compression algorithm finds repetitive patterns and regularities in the original sequence, i.e. the original sequence is highly ordered. If the compressed sequence is long then the original sequence represents a chaotic piece of information, i.e., a high entropy environment is detected. Depending on whether A_1 or A_2 are applied, some kinds of patterns will be detected as repetitive while some others will not (actually, there does not exist any perfect compression strategy). Hence, if a general approach is considered then the information entropy is a relative notion indeed.

In this paper we present a formal framework to detect living beings in information streams. Following the Maximum Entropy Production Principle, we seek for low entropy structures that increase the entropy around them. Compression algorithms are used to define *order* and *chaos* in each case. In fact, the proposed method to detect life is parameterized by the definition of entropy we wish to consider, i.e. by the specific *compression algorithm* we are considering. Several formal notions to detect life and classify it, as well as to assess the *biodiversity* of the analyzed environment, are considered. In addition, a toy example is considered and some experimental results are reported. In particular, we search for structures fulfilling our definition of life within a video stream representing an execution of the classical *Snake* game.

The rest of the paper is structured as follows. In the next section we present some preliminary concepts and we use them to define Life in terms of compression rates according to the Maximum Entropy Production Principle. Besides, we present some concepts concerning the biodiversity of artificial ecosystems, and we deal with the notions of births and deaths in our framework. In Section 3, we present an example where we apply some of the proposed concepts to detect life in a game execution. Finally, some conclusions and future work are given in Section 4.

2 Formal Model

In this section we present some basic notions to define and manipulate information in our framework. We denote by *world* the information source where we search for living structures. In formal terms, a world is a set of points located in the space and time where each point has attached a binary value. We represent these sets by means of a function, as shown below.

Definition 1. An $n-$*world* is a partial function $w : \mathbf{N}^n - \rightarrow \{0,1\}$ with finite domain. We say that the *scope* of w is the domain of w, and we denote it by S_w.

We denote by `n-Worlds` the set of all possible $n-$worlds. □

Let us note that the previous definition uses n dimensions without making any special difference to represent the time. In fact, we may assume that the time is just one of the n dimensions. Thus, we can trivially represent *dynamic* worlds evolving in time. Next we define the parts of a world. This notion will be required later to identify living structures within a world. In addition, we represent some algebraic operators that will be used to combine worlds.

Definition 2. Let w, w' be two $n-$worlds. We say that w' is a *subworld* of w, denoted by $w' \subseteq w$, if $S_{w'} \subseteq S_w$ and for all $x \in S_{w'}$ we have $w'(x) = w(x)$.

Let w_1, w_2 be two $n-$worlds such that $S_{w_1} \cap S_{w_2} = \emptyset$. The *union* of worlds w_1 and w_2, denoted by $w_1 \cup w_2$, is a new world w with scope $S_w = S_{w_1} \cup S_{w_2}$ such that $w(x) = w_1(x)$ if $x \in S_{w_1}$ and $w(x) = w_2(x)$ if $x \in S_{w_2}$. □

Once we can deal with subparts of a world, we can present some preliminary notions to identify repetitive patterns inside it. Since living beings are parts of the world where they exist, we can use subworlds to delimit those parts of the world that actually denote living structures.

Let us note that if a given structure appears several times then we can *codify* the presence of all its instances with a representation shorter than if these structures were different. Thus, repetitive patterns allow to reduce the length of the codification of the whole world where they are. If we consider this argument the other way around, repetitive patterns can be identified in a world by applying a compression algorithm to the world. Essentially, a compression strategy is just a codification. That is, it is a transformation of a world into a sequence of bits. These transformations induce a *compression rate*, that is, the rate between the length of the compressed sequence of bits and the size of the world.

Definition 3. A *compression strategy* for $n-$worlds is a function C where we have $C : \texttt{n-Worlds} \rightarrow \{0,1\}^*$.

Let $w \in \texttt{n-Worlds}$. The *compression rate* of C for w is defined as $\frac{\texttt{length}(C(w))}{\mid S_w \mid}$, and it is denoted by $\texttt{CompRate}(C, w)$. □

Let us note that we are considering a very general notion of compression strategy. For instance, if we restricted ourselves to e.g. Huffman codes [4] or algorithms such as LZW [14] then the generality of the framework would be reduced. In contrast, searching for redundancies in several different ways is allowed in the proposed framework. For instance, we may search for decompositions of frequencies by using the discrete Fourier transformation or the discrete cosine transformation (see e.g. [6,10]) (in particular, the JPEG transformation will be considered in the example presented at the end of the paper). The generality of the previous definition will allow us to search for life patterns in a broader sense than usual. In particular, the criterion to detect repetitive patterns will depend on the particular compression strategy considered in each situation. Thus, a pattern under a certain strategy could not be a pattern under another one. This reflects the fact that the interpretation given to the world strongly depends on the *rational model* we use to describe that world. In our case, the considered rational model is denoted by the compression strategy.

Next we define what a (living or not) repetitive pattern is. A *pattern* is a subworld such that, when it is considered as part of a given world, the compression rate of the world is reduced. In other words, a pattern allows to increment the order of the world where it is inside. A subworld can be a pattern due to two different reasons. On the one hand, a subworld can help to form redundancies in the world it belongs to. In this case, we say that it is an *exogenous* pattern. On the other hand, a subworld can be a pattern because it has many internal redundancies, in which case we say that it is an *endogenous* pattern. In order to make such distinction, we take into account the compression rate of the subworld as if it were isolated indeed.

Definition 4. Let w, w_1, w_2 be $n-$worlds such that $w = w_1 \cup w_2$. We say that w_1 is a *pattern* in w under compression strategy C if $\texttt{CompRate}(C, w) < \texttt{CompRate}(C, w_2)$.

Besides, if $\texttt{CompRate}(C, w_1) \geq \texttt{CompRate}(C, w_2)$ then we say that w_1 is an *exogenous* pattern; if $\texttt{CompRate}(C, w_1) < \texttt{CompRate}(C, w_2)$ then we say that w_1 is an *endogenous* pattern. □

The previous definition does not imply that patterns have a *low* entropy level, but that their entropy is low in *relation* with the entropy of the world where they exist. In fact, even if a pattern is endogenous, it is not guaranteed that its entropy level is low, as the rate is measured by taking into account the world it belongs to.

Patterns can be nested, that is, we can find life inside living entities. By introducing this concept, we can manage notions such as cooperative living subentities constituting global living entities. We assume that the same compression strategies are considered at both nesting levels.

Definition 5. Let w_1 be a pattern in w under compression strategy C and let w_2 be a pattern in w_1 under compression strategy C. Then, we say that w_2 is a *subpattern* of w_1. □

Let us consider the notions of entropy and life. Intuitively, and following the ideas shown in [9], a living creature is a structure that maintains low entropy inside it, while increasing the entropy of the environment surrounding it. Thus, in order to decide whether a pattern is a living pattern or not, we have to compare the entropy of the pattern with that of its surroundings. We define the entropy of a subworld as the ratio between the entropy of that subworld and that of the world it belongs to. Next we introduce a notion of pattern which is parameterized by an entropy threshold. It allows us to compare patterns in terms of their relative level of order with respect to their world.

Definition 6. Let w, w' be two worlds such that w' is a subworld of w. The *entropy level* of w' in w under compression strategy C, denoted by $\texttt{Entropy}(C, w', w)$, is given by $\frac{\texttt{CompRate}(C,w')}{\texttt{CompRate}(C,w)}$.

We say that w' is an $\alpha-$*ordered pattern* of w under compression strategy C if w' is a pattern of w under compression strategy C and $\texttt{Entropy}(C, w', w) < \alpha$ for a given constant α. □

Even if a pattern is ordered inside its world, this does not imply that the pattern is a living pattern. We must also take into account that living entities must *increase* the entropy around them. Thus, we need to detect if there exists an *evolution* towards higher entropy. Reasoning about how some parameter evolves requires to identify a dimension of the world (or linear combination of them) as the *time* dimension, i.e. we have to define what is the direction of the evolution. Next we define sequences of increasing entropy. We say that a pattern is *alive* if it keeps a low entropy level along the evolution of its world and, simultaneously, the entropy level of its world increases along time.

Definition 7. Let $w_1, \ldots, w_n, w$ be worlds such that $w = \bigcup_{i=1}^{n} w_i$. We say that the sequence $w_1 \cdots w_n$ is an *evolution of entropy* under compression strategy C if for any i, j with $1 \leq i < j \leq n$ we have $\texttt{CompRate}(C, w_i) < \texttt{CompRate}(C, w_j)$.

Let w, w', w'' be worlds such that $w = w' \cup w''$ and w' is an $\alpha-$ordered pattern of w under compression strategy C. Let $w_1 \cdots w_n$ with $w = \bigcup_{i=1}^{n} w_i$ be an evolution of entropy under compression strategy C. We say that w' is an $\alpha-$*living pattern* across $w_1 \cdots w_n$ under C if there exist two sequences $w'_1 \cdots w'_n$ with $w' = \bigcup_{i=1}^{n} w'_i$ and $w''_1 \cdots w''_n$ with $w'' = \bigcup_{i=1}^{n} w''_i$, such that $w''_1 \cdots w''_n$ is an evolution of entropy under C and for all $1 \leq i \leq n$ we have that $w_i = w'_i \cup w''_i$ and w'_i is an $\alpha-$ordered pattern of w_i under C. □

It is worth to point out that $w'_1 \cdots w'_n$ (that is, the sequence representing the evolution of the living entity) could also be an evolution of entropy. That is, the internal entropy of an alive creature could also be increasing, provided that it is still a pattern inside its world. Intuitively, this implies that the tendency of the world towards chaos must be *faster* than the tendency of the living entity itself.

As we said before, the evolution of the entropy is not constrained to follow a specific direction. Since there are different ways to split a world into scenes, there exist several possible interpretations of time, and all notions depend on this choice. This increases the generality of the proposed framework. Let us remark that we are dealing with *information*, so our definition must be independent of the possible transformations being applied to such information. For instance, let us suppose that the world represents a video stream. Each temporal frame of the video could be located in a different part of the x axis of the information stream (e.g., a file). The evolution of the video over time is codified by locating each frame in a specific physical area of the stream. Hence, a flexible way to identify the time dimension must be provided.

It is worth to point out that the previous definition does take into account one of the factors considered critical for identifying life in terms of the Maximum Entropy Production Principle [9]. According to this principle, living creatures *generate* entropy in their surroundings. That is, they are the *reason* of the increment of entropy. In our approach, we detect life by just *observing* information, that is, we do not interact with it. Hence, we do not have the capability of changing the observing environment, which would allow us to check an alternative scenario where the creature does not exist. This would allow us to compare the evolution of the entropy in both cases, which is required to determine if the existence of the creature causes it. Studying the case where it is possible to interact with the analyzed environment is out of the scope of this paper and is left as future research.

Once we have proposed our notion of life, we can use it to define some higher level concepts. Next we consider the notion of *subliving patterns*. A subliving pattern is a living entity inside another living entity. In particular, the world of a subliving entity is the living entity it belongs to.

Definition 8. Let w' be an $\alpha-$living pattern across $w_1 \cdots w_n$ under the compression strategy C, and let w'' be an $\alpha-$living pattern across $w'_1 \cdots w'_n$ under the compression strategy C such that $w' = \bigcup_{i=1}^{n} w'_i$ and for all $1 \leq i \leq n$ we have $w'_i \subseteq w_i$. Then, we say that w'' is an *$\alpha-$subliving pattern* of w' across $w_1 \cdots w_n$ under C. We say that w' is a *fully $\alpha-$living pattern* across $w_1 \cdots w_n$ if there exist m worlds $w'_1, \ldots, w'_m$ $(m \geq 2)$ such that $w' = \bigcup_{i=1}^{m} w'_i$ and for all $1 \leq i \leq m$ we have that w'_i is an $\alpha-$subliving pattern of w' across the evolution $w_1 \cdots w_n$. □

As stated in [13], the *biological diversity* is the variety and variability among living organisms and the ecological environments in which they occur. Thus, the *diversity* can be defined as the number of different items and their relative frequency. In order to calculate the biodiversity of a world in our framework, we have to consider the life existing in it. Nevertheless, since diversity is required, the biodiversity does not increase by considering very similar living beings. On the contrary, the diversity is high only if it is possible to find a subset of the world such that its diversity is high. This subset should be defined in such a way that its members are *canonical representatives* of the different models of life appearing in the ecosystem. Then, the biodiversity will be calculated by considering two factors: The internal diversity of the subset (which indicates that present models are different among them) and its size (which indicates the amount of diverse life in the ecosystem). Hence, we calculate the biodiversity of a world by selecting the set that maximizes both factors together.

Definition 9. Let $w_1 \cdots w_n$ be an evolution of entropy under C, and let $w = \bigcup_{i=1}^{n} w_i$. The *$\alpha-$biodiversity* of $w_1 \cdots w_n$ under the compression strategy C is defined as:

$$\max \left\{ \frac{\mid S_{w'} \mid}{\mid S_w \mid} \cdot \texttt{CompRate}(C, w') \,\middle|\, w' = \bigcup_{i=1}^{m} w'_i \;\wedge\; \forall\, 1 \leq i \leq m : w'_i \in L \right\}$$

where L denotes the set of all $\alpha-$living patterns across $w_1 \cdots w_n$ under the compression strategy C. □

In the previous definition, we search the set of living beings such that, considering this set as a whole, the compression rate is the highest (which indicates that the diversity is high). At the same time, we search for the set whose size is as closer as possible to the size of the whole world (which indicates that the amount of diverse life is high). The multiplication of both factors provides our measure of biodiversity. Let us remark that the biodiversity is monotonic non-decreasing with respect to α. This is because higher values of α increase the freedom to choose living patterns, which allows to maximize the biodiversity value. In particular, those sets we can consider with a lower α can also be selected with a higher one.

The proposed formal framework also allows to define the notions such as *births* and *deaths* for living entities. For the sake of clarity, in previous definitions we assumed that each alive entity is alive during the *whole* considered period (i.e., during the considered evolution of entropy). Nevertheless, we can extend the previous concepts to deal with a more general situation where creatures are born and die.

In the following definition we introduce the concepts of birth and death. Let us remark that both concepts are relative to the entropy level α required in each case to determine if patterns are alive.

Definition 10. Let $w_1 \cdots w_m$ be an evolution of entropy under the compression strategy C and let w' be an $\alpha-$living pattern across $w_k \cdots w_n$ under the compression strategy C, where $1 \leq k \leq n \leq m$. Finally, let $w = \bigcup_{i=k}^{n} w_i$.

We say that the $\alpha-$*birth date* of w' is w_k if there does not exist $w'' \subseteq w_{k-1}$ such that w'' is an $\alpha-$ordered pattern of w_{k-1} under the compression strategy C and $w' \cup w''$ is an $\alpha-$ordered pattern of $w_{k-1} \cup w$ under the compression strategy C.

We say that the $\alpha-$*death date* of w' is w_n if there does not exist $w'' \subseteq w_{n+1}$ such that w'' is an $\alpha-$ordered pattern of w_{n+1} under the compression strategy C and $w' \cup w''$ is an $\alpha-$ordered pattern of $w_{n+1} \cup w$ under the compression strategy C. □

The intuitive idea behind the dates of birth and death is that they are dates such that it is not possible to extend the life of the creature after its death or before its birth.

3 Experiments

In this section we present an example of the framework presented in this paper by using a classical software game. This game is *Snake*. Essentially, the goal of this game consists in making the snake to grow up as much as possible by eating all the food it finds in the world. The snake dies either if it crashes against a part of its own body or against one of the walls surrounding the world. In addition to the original rules of the classic game, we introduce a new concept that will be necessary to deal with the proposed notion of life: Rubbish. When the snake eats something, it randomly produces rubbish in the surroundings next to it. This simulates the degradation of the environment caused by life. Since our notion of life requires that the entropy of the environment grows along time, a kind of degradation will be required to find life in this system.

We represent an execution of this game by means of a *world*. According to Definitions 1, 2, and 3, we consider a 3-world (2 spacial dimensions plus the *time*) where the size of each spacial dimension is 512. A *frame* of the scene (that is, the information of the world for a specific time) is shown in Figure 3 (left). The food is shown in blue color and the snake is green. In the following, we will use w to represent a subworld denoting a single frame.

Living structures are images moving across the screen along time, so we must be able to systematically search for parts of the image to be considered as possible living structures. In order to do it, we present an algorithm that automatically considers different ways to split the screen into pieces of different size and assesses the suitability of each piece to denote a living structure. The main part of the algorithm is depicted in the adjacent figure. This heuristic greedy algorithm looks for a square whose size is a divisor of n, being n the length of each spatial dimension. The cost of the algorithm is $\mathcal{O}(log(n))$. Intuitively, the algorithm works as follows: First, we split the image into four quadrants.

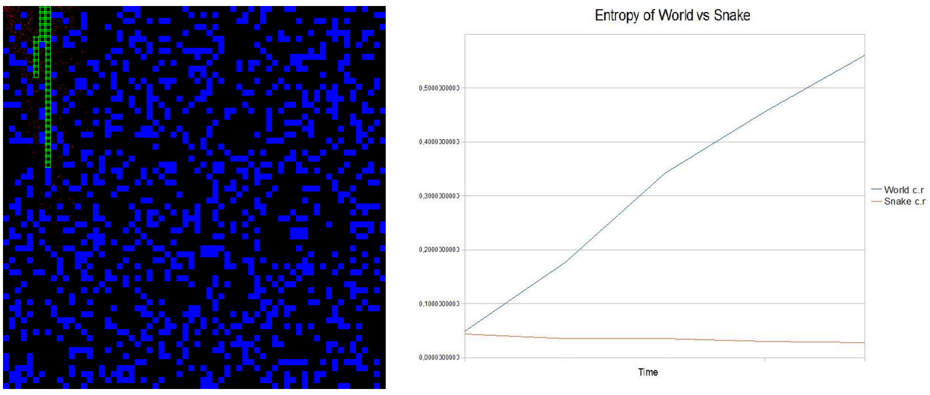


Fig. 1. A world representing an image (left) and the evolution of compression rate (c.r.) of the world and the snake (right)

We choose the quadrant where the conditions required to find life are best suited (that is, the square has low entropy but, at the same time, the entropy evolves along time in the rest of the frame). The function `BestComp` abstracts the criterion used to make this selection. Next, this quadrant is split again into four quadrants, and so on. When the process finishes, that is, when the minimal size is reached, the best square considered so far (regardless of its size) is identified. For the sake of clarity, some subsequent operations of the algorithm are not depicted in the figure. In order to *extend* the best square in some directions, other squares adjacent to it are considered. If the figure resulting by adding an adjacent square is better suited, then we take it as new best figure, and we repeat the same process for some additional turns. In this way, more complex forms (not just squares) can be formed. More formally, given a subworld w denoting a frame, the algorithm chooses two different subworlds w_1 and w_2, that is, subsets of w, fulfilling the following conditions. The first subworld, w_1, is the rectangular area chosen by applying the algorithm depicted in Figure 2. We define w_2 as the the rest of the frame, that is, we have $w = w_1 \cup w_2$ with $S_{w_1} \cap S_{w_2} = \emptyset$.

As we said before, we consider that the entropy of a subworld is the ratio between the compression rate of that subworld and that of the world it belongs to. So, our notion of entropy is a notion of *relative* order between a subworld and the world this subworld is inside. Following this idea, we perform an experiment to determine whether the snake should be considered an (artificial) alive creature according to the proposed notions. We use the JPEG compression algorithm to measure the entropy along time: Each individual frame is compressed by using this algorithm, and the evolution of resulting compression rates are considered. As we said before, other compression algorithms lead to different implicit definitions of what should be considered an ordered pattern and what should not.

In Figure 3 (right) we can observe the evolution of the entropy along the time of w_1 (the snake) and w (the world). Since the snake represents a simple repetitive pattern, the complexity of its JPEG codifications (that is, the *size* of the compressed images representing its frames along time) do not significantly increase along time. On the contrary,

```
input : An n-world represented by a Bitmap Matrix $H$ of size $n \times n$.
output: A bitmap denoting a good life candidate within the world.

$n \leftarrow (n\ DIV\ 2) \times 2$ ;
$size \leftarrow \frac{n}{2}$;
$B \leftarrow$ MAXINT; $left \leftarrow 1$; $right \leftarrow n$; $up \leftarrow 1$; $down \leftarrow n$;
while (size $\geq 1$) do
    $left_{new} \leftarrow left$;
    $right_{new} \leftarrow right$;
    $up_{new} \leftarrow up$;
    $down_{new} \leftarrow down$;
    if ($B \geq$ BestComp($H$,$left$,$\frac{right}{2}$, $up$, $\frac{down}{2}$)) then
        $left_{new} \leftarrow left$;
        $right_{new} \leftarrow \frac{right}{2}$;
        $up_{new} \leftarrow up$;
        $down_{new} \leftarrow \frac{down}{2}$;
        $B \leftarrow$ BestComp($H$,$left$,$\frac{right}{2}$, $up$, $\frac{down}{2}$);
    end
    if ($B \geq$ BestComp($H$, $\frac{right}{2}$,$right$, $up$, $\frac{down}{2}$)) then
        $left_{new} \leftarrow \frac{right}{2}$;
        $right_{new} \leftarrow right$;
        $up_{new} \leftarrow up$;
        $down_{new} \leftarrow \frac{down}{2}$;
        $B \leftarrow$ BestComp($H$, $\frac{right}{2}$,$right$, $up$, $\frac{down}{2}$);
    end
    (... continue);
end
```

```
input : An n-world represented by a Bitmap Matrix $H$ of size $n \times n$.
output: A bitmap denoting a good life candidate within the world.

$n \leftarrow (n\ DIV\ 2) \times 2$ ;
$size \leftarrow \frac{n}{2}$;
$B \leftarrow$ MAXINT; $left \leftarrow 1$; $right \leftarrow n$; $up \leftarrow 1$; $down \leftarrow n$;
while (size $\geq 1$) do
    (... continue);
    if ($B \geq$ BestComp($H$, $\frac{right}{2}$,$right$, $\frac{down}{2}$, $down$)) then
        $left_{new} \leftarrow \frac{right}{2}$;
        $right_{new} \leftarrow right$;
        $up_{new} \leftarrow \frac{down}{2}$;
        $down_{new} \leftarrow down$;
        $B \leftarrow$ BestComp($H$, $\frac{right}{2}$,$right$, $\frac{down}{2}$, $down$);
    end
    if ($B \geq$ BestComp($H$,$left$,$\frac{right}{2}$, $\frac{down}{2}$, $down$)) then
        $left_{new} \leftarrow left$;
        $right_{new} \leftarrow \frac{right}{2}$;
        $up_{new} \leftarrow \frac{down}{2}$;
        $down_{new} \leftarrow down$;
    end
    $left \leftarrow left_{new}$;
    $right \leftarrow right_{new}$;
    $up \leftarrow up_{new}$;
    $down \leftarrow down_{new}$;
    $size \leftarrow \frac{size}{2}$;
end
```

Fig. 2. Searching good life candidate in an image

the JPEG codifications of frames representing the world become longer as time passes. Since the snake increases the rubbish every time it eats, the amount of rubbish increases along time, and representing this information in the compressed format requires more bits. Since the entropy of the snake remains low and the entropy of its world increases along time, we can conclude that the snake represents an alive pattern according to the notions presented in previous sections. Let us note that, as we have already commented before, we need that the amount of rubbish increases as the snake eats and grows. Otherwise, the world would reduce its entropy along time. Let us note that the compression rate of an *empty* world is better (uniformly colored areas are easier to compress). Thus, if the rubbish were not generated then the world of the snake would not be globally tend towards chaos according to the selected compression strategy.

4 Conclusions and Future Work

In this paper we have presented a formal framework to identify living entities inside an abstract information environment. Following the Maximum Entropy Production Principle, the proposed method is based on the analysis of the entropy of the components of the system. More precisely, we have compared the entropy of entities with the evolution of the entropy of the world they belong to. The entropy is measured in terms of *compression rates*. This allows us to measure the order degree of some information in a computational environment. Other related concepts, including notions such as death, biodiversity, or biologic families, have been discussed. We have illustrated the proposed concepts with a toy example where a living entity is detected in a simple game execution. Since the compression rate degrades but, simultaneously, the analyzed pattern remains ordered along time, we have concluded that this entity constitutes an alive entity according to the Maximum Entropy Production Principle.

As future work, we want to apply the proposed formal framework to analyze the presence of life in classical artificial environments. In particular, we wish to compare our definition of Life with Class IV considered by [15] and the Lambda metric proposed by [7] in the specific context of Cellular Automata. Besides, we wish to define alternative life detection notions. Contrarily to the formal notions presented in this paper, which are based on the simple *observation* of the environment, we wish to consider an alternative framework where we could extract conclusions by *interacting* with the analyzed environment.

Acknowledgments. We would like to thank Manuel Núñez and Natalia López for their interesting suggestions, as well as the anonymous reviewers for their valuable comments.

References

1. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning Ways for Your Mathematical Plays. Academic Press, London (1982)
2. Bruers, S.: A discussion on maximum entropy production and information theory. Journal of Physics A: Mathematical and Theoretical 40(27), 7441–7450 (2007)
3. Dewar, R.: Information theory explanation of the fluctuation theorem, maximum entropy production and self-organized criticality in non-equilibrium stationary states. Journal of Physics A: Mathematical and General 36(3), 631–641 (2003)
4. Huffman, D.A.: A method for the construction of minimum redundancy codes. Proceedings of the Institute of Radio Engineers 40(9), 1098–1101 (1952)
5. Kim, K.-J., Cho, S.-B.: A comprehensive overview of the applications of artificial life. Artificial Life 12(1), 153–182 (2006)
6. Kok, C.W.: Fast algorithms for computing discrete cosine transform. IEEE Transactions on Signal Processing 45, 757–760 (1997)
7. Langton, C.: Studying artificial life with cellular automata. Physica D 22, 120–149 (1986)
8. Langton, C.: Artificial Life: An Overview. MIT Press, Cambridge (1995)
9. Prigogine, I., Stengers, I.: Order Out of Chaos. Bantam Books (1984)
10. Rafiei, D., Mendelzon, A.: Efficient retrieval of similar time sequences using DFT. In: 5th International Conference on Foundations of Data Organization and Algorithms (FODO 1998) (November 1998)
11. Schneider, E.D., Kay, J.J.: Life as a manifestation of the second law of thermodynamics. Mathematical and Computer Modelling 19(6-8), 25–48 (1994)
12. Shannon, C.E.: A mathematical theory of communication. Bell Systems Technical Journal 27, 379–423 (1948)

13. Office of Technology Assessment U.S. Congress. Technologies to mantain biological diversity, OTA-F-330. U.S. Goverment Printing Office, Washington (March 1987)
14. Welch, T.A.: A technique for high-performance data compression. IEEE Computer 17(6), 8–19 (1984)
15. Wolfram, S.: Cellular automata. Los Alamos Science 9, 2–21 (1983)
16. Wolfram, S.: Cellular Automata and Complexity. Addison-Wesley, Reading (1994)

How a Generative Encoding Fares as Problem-Regularity Decreases

Jeff Clune[1], Charles Ofria[1], and Robert T. Pennock[1,2]

[1] Department of Computer Science and Engineering,
[2] Department of Philosophy & Lyman Briggs College
Michigan State University, East Lansing, MI, USA 48824
{jclune,ofria,pennock5}@msu.edu

Abstract. It has been shown that generative representations, which allow the reuse of code, perform well on problems with high regularity (i.e. where a phenotypic motif must be repeated many times). To date, however, generative representations have not been tested on irregular problems. It is unknown how they will fare on problems with intermediate and low amounts of regularity. This paper compares a generative representation to a direct representation on problems that range from having multiple types of regularity to one that is completely irregular. As the regularity of the problem decreases, the performance of the generative representation degrades to, and then underperforms, the direct encoding. The degradation is not linear, however, yet tends to be consistent for different types of problem regularity. Furthermore, if the regularity of each type is sufficiently high, the generative encoding can simultaneously exploit different types of regularities.

Keywords: Evolution, regularity, modularity, ANN, NEAT, HyperNEAT.

1 Introduction

While the field of evolutionary computation has produced impressive results, the complexity of its evolved solutions pales in comparison to organisms in the natural world. One of several likely reasons for this difference is that evolutionary computation typically uses a *direct encoding* (also known as *direct representation*), where, relative to some environment E, every part of the phenotype is coded for separately in the genome. Given that natural organisms can contain trillions of parts (e.g. cells in the human body), a direct representation of such an organism would require a genome with at least that many separate genetic elements. We do not find such inefficient genomes in nature. An alternative is a *generative encoding* (or *generative representation*), where, relative to E, elements in a genome can be reused to produce many parts of a phenotype [1-9]. For example, about 25,000 genes encode the information that produces the trillions of parts that make up a human [10]. Generative encodings allow evolution to search a genotype space with far fewer dimensions than that of the final phenotype. Further benefits of generative encodings are that the reuse of code facilitates the evolution of modular phenotypes and that mutations can produce coordinated phenotypic effects (e.g. one mutation lengthening all legs by the same amount).

Previous researchers have found that generative encodings produce more modular, complex phenotypes, higher fitnesses, and more beneficial mutations on average than direct encoding controls [1].

While it has been shown that generative encodings can outperform direct encodings [1-7], in every case the authors of this paper are aware of, the problem was nearly perfectly regular or the regularity of the problem was unspecified and ambiguous. Gruau's work evolving neural nets with the generative encoding called Cellular Encoding used problem domains of bit parity or bit symmetry [4], which are both highly regular, or pole-balancing [5], where the regularity of the problem is unknown. Even for the pole-balancing problem, however, there is left-right symmetry and just a few tasks (e.g calculating velocity) need to be repeated many times. Hornby [1] demonstrated that a generative encoding based on L-systems outperformed a direct encoding control when applied to the perfectly regular parity problem and to evolving tables and mobile creatures (where repeating similar leg modules gave huge fitness gains). The regularity of the Nothello game from [7] is unknown. Recently, a new generative encoding, called Hypercube-based NEAT (HyperNEAT), has been shown to outcompete a direct encoding control on two problems that require the repetition of the same network motif [2,3].

These works show that generative encodings do well on highly regular problems, but they raise the question of whether generative encodings achieve their increased performance in regular problem domains at the cost of performing poorly in irregular problem domains. For example, how good are generative encodings at producing an exception to the rule? Furthermore, do they provide advantages for low and intermediate amounts of regularity, or only when regularity is high? What is needed are tests of generative versus direct encodings on problems that allow us to explicitly vary *only* their regularity. Such investigations are conducted in this paper.

2 The Experimental System

This study uses a generative encoding that evolves neural nets, which is one of the common uses for generative encodings within the field of evolutionary computation [1-8]. HyperNEAT [2,3] was recently introduced as a generative representation that can evolve neural nets using the principles of the widely used NeuroEvolution of Augmenting Topologies (NEAT) algorithm [11]. HyperNEAT evolves Compositional Pattern Producing Networks (CPPNs), each of which is a function that takes an input and produces an output. The inputs to the CPPN function are a constant bias value and the locations on a Cartesian grid of both an input node (e.g. $<x_1=4, y_1=4>$) and an output node (e.g $<x_2=5, y_2=5>$). The function takes these five values (bias, x_1, y_1, x_2, y_2) as input and produces an output value that determines the weight of the link between the input and output node. All pairwise combinations of input and output nodes are iteratively passed as inputs to a given CPPN to determine what the weight value is between each input node and each output node. Thus the CPPN function is a genome that encodes for a neural network phenotype.

Evolution proceeds in HyperNEAT by evolving a population of CPPN functions. Each CPPN is itself a directed graph network where each node is a math function comprised of the following functions: sine, sigmoid, cosine, Gaussian, square, absolute root, linear, or one's complement. The nature of the functions used can create a wide variety of desirable properties, such as symmetry (e.g. an absolute value or Gaussian function) and repetition (e.g. a sine or cosine function) that evolution can take advantage of. That a directed network of functions is used allows nested coordinate frames to develop, such that, for instance, a sine function used early in the network can create a repeating theme that, when passed into the symmetrical absolute value function, creates a repeating series of symmetrical motifs. This is similar to how natural organisms develop. For example, many organisms set up a repeating coordinate frame (e.g. body segments) within which are symmetrical coordinate frames (e.g. left-right body symmetry). The links between each node in a CPPN have a weight value that can magnify or diminish the values that pass along them. The ability to change these weights enables evolution to, for example, give strong weight to one part of the network generating symmetry while rendering the influence of another aspect of the network more subtle. When CPPNs are evolved artificially with humans doing the selection, the evolved shapes look surprisingly beautiful, complex and natural [9]. More importantly, they exhibit the desirable features of generative encodings, namely, the repetition of themes, symmetries and hierarchies, with and without variation.

Variation in HyperNEAT occurs when mutations change the CPPN function networks. Mutations can add a node to the graph, which results in the addition of a function to the network, or change the weights of links within the network. The evolution of the population of CPPN networks occurs according to the principles of NEAT, which was originally designed to evolve neural nets. NEAT can be fruitfully applied to CPPNs because the population of CPPN networks is similar in structure to a population of neural networks.

The NEAT algorithm is unique in three main ways [11]. Initially, it starts with small genomes that encode simple networks and slowly 'complexifies' them via mutations that add nodes and links to the network. This complexification enables the algorithm to evolve the network topology in addition to its weights. Secondly, it uses a fitness sharing mechanism that preserves diversity in the system and allows new innovations time to be tuned by evolution before forcing them to compete against rivals that have had more time to mature. Finally, it uses historical information to perform crossover in a way that is effective yet avoids the need for expensive topological analysis. A full explanation of HyperNEAT [2, 3] and NEAT [11] can be found elsewhere.

It is helpful that a good direct encoding version of NEAT exists that can serve as a control. Perception NEAT (P-NEAT), so named because it evolves a series of perceptrons, has been previously used to compare the generative encoding of HyperNEAT with a direct encoding that is similar to HyperNEAT in all ways, save its use of the generative CPPNs [2, 3]. P-NEAT directly evolves the neural net phenotypes. It is the same as NEAT without the complexification. In other words, P-NEAT uses evolution to tune the weights of a network with a fixed topology. Since in HyperNEAT the complexification is performed on the CPPN, but the resultant neural network topology remains fixed, the topology of the P-NEAT neural network is also fixed. The rest of the elements from NEAT (e.g. fitness sharing) remain the same between HyperNEAT and P-NEAT, making the latter a good control.

Following Gauci and Stanley (2007), a configuration is used that separates the inputs and outputs onto two separate planes. This configuration features a two dimensional, n-by-n Cartesian grid of inputs and a corresponding n-by-n grid of outputs. There are no hidden nodes and recurrence is disabled, so each of the n^2 inputs nodes has a link of a given weight to each of the n^2 output nodes (although weights can be zero, functionally eliminating the link). The parameter configurations for HyperNEAT and PNEAT are the same as in Gauci and Stanley (2007), except that the probability of adding a link was 30% (up from 10%) and the MutationPower, which controls the magnitude of weight mutations, was lowered from 2.5 to 0.1 after preliminary experiments revealed that this improved performance in the problem domains used in this paper. Every experimental trial within a treatment differed only in its random number generator seed, which influenced stochastic events such as mutations and the creation of randomized targets (either maps or weights). HyperNEAT and P-NEAT trials with the same random number seed in each treatment had the same targets.

3 Problem Domains and Results

The first problem, which will be called 'Bit Mirroring,' is intuitively simple yet provides multiple types of regularities, each of which can be scaled independently. For each input a target output is assigned (e.g. the input <x_1=3, y_1=3> could be paired with output <x_2=7, y_2=7>). 1s and -1s ones are randomly provided to each input, and the fitness of an organism is incremented if that one or negative one is reflected in the target output. Outputs greater than zero are considered a one, and values less than or equal to zero are considered a negative one. To reduce the effect of the randomness of the inputs, every generation each organism is evaluated on ten different sets of random inputs and these scores are summed to produce a fitness score for that organism. The max fitness is thus n^2 (one potential right answer for each input) x 10.

The correct wiring is to create a positive valued link between each input node and its target output and, importantly, to zero out all links between each input node and non-target output nodes. That there is a correct wiring motif that needs to be repeated for each input cell creates an 'inherent regularity' to the problem. However, this inherent regularity is constant for a given grid size. The Bit Mirroring problem is challenging for evolutionary algorithms because links between input nodes and non-target nodes are likely to exist in initial random configurations, crop up through mutation, and can complicate fitness landscapes. Imagine, for example, that a mutation switches the weight value on a link between an input node and its target output from zero to a positive number. The organism is now closer to the ideal wiring, but it may not receive a fitness boost if other incorrect links to that output node result in the wrong net output. While the problem is intuitively simple, it is not trivial.

Recall that highly regular problems are those where one motif must be repeated multiple times. One simple way to construct a highly regular Bit Mirroring problem is for the x and y values of the input and output nodes to be the same (i.e. the target is directly across). For example, <x_1=5, y_1=6> should connect to <x_2=5, y_2=6>. There are at least three types of regularity in this highly regular version of the Bit Mirroring problem: 1) inputs and output targets have the same x values (they are in the same column), 2) inputs and output targets have the same y values (they are in the same row), and 3) the inherent regularity in the Bit Mirroring problem (see above). Each type of regularity can be scaled from high to low.

The first experiment uses a 7x7 grid and decreases the 'within-column' regularity by reducing the percentage of inputs whose target is constrained to be in the same column. Unconstrained nodes must have the same y value, but can have a different x value. 10 trials were performed for each treatment lasting 2000 generations. Fig. 1a reveals that the performance of HyperNEAT falls off as within-column regularity decreases. HyperNEAT is able to perfectly solve the problem in all but two trials of the most regular treatment, when the targets are all constrained to be directly across. As the within-column regularity decreases, the performance of HyperNEAT falls off fast. Interestingly, HyperNEAT does not benefit from the within-column regularity when 50% or fewer of its nodes are regularized in this way (only treatments with 60% or more column-constrained targets have fitnesses significantly better at a p<.05 level than fitness values from the treatment with 0% of nodes column-constrained; this and all future p values use Matlab's Mann-Whitney U-test).

The second experiment, which also involved 10 trials lasting 2000 generations and a 7x7 grid, scales a similar but different type of regularity by allowing all targets to be random with respect to column, but decreasing the percent that are constrained to be in the same row (Fig. 1b). In a sense, experiment two picks up where experiment one left off. In fact, the least regular treatment from experiment one and the most regular treatment from experiment two have identical constraints (although different randomly generated mappings), which is why their distributions are similar. The performance of HyperNEAT also decreases as this type of regularity is diminished. Surprisingly, the pattern of degradation is similar to experiment one; HyperNEAT no longer provides a fitness boost due to within-row regularity once that regularity falls below 60% (p only <.05 comparing 0% row-constrained to >=60% row-constrained treatments). While it is possible that running experiment one and two longer would have allowed significant differences to develop between less-regular treatments, it is relevant that no significant difference was present after 2000 generations, which is a substantial number in the field of evolving neural nets. It is interesting that the *range* of fitness values is also correlated with the regularity of the problem for HyperNEAT. This might be because, when regularity is present, the generative representation either discovered and exploited it, which would result in high fitness values, or it failed to fully discover the regularity, at which point its fitness more closely resembles less regular treatments.

Experiments one and two were also performed using P-NEAT, the direct representation control for HyperNEAT. As expected, the fitnesses produced by P-NEAT were not affected by the regularity of the problem. While for space constraints we only show P-NEAT values from experiment two (Fig. 1c), none of the P-NEAT treatments from experiment one were significantly different from P-NEAT treatments from experiment two (p< .05). Furthermore, within both experiments, none of the treatments were significantly different than that experiment's 0% constrained treatment (p >.05). All HyperNEAT treatments from experiment one do significantly better than P-NEAT treatments from experiment one, due to both the within-row regularity present throughout and the inherent regularity of the Bit Mirroring problem. In experiment two, the within-row regularity decreases, leaving only the inherent regularity. However, presumably due to the inherent regularity of the problem, HyperNEAT still significantly outperforms P-NEAT on all treatment conditions except for the 20% constrained treatment. Computational constraints prevented the performance of more trials, which may have eliminated this overlap in performance on the 20% constrained trial. While it is possible that running the trials for more generations would have allowed P-NEAT to catch up to HyperNEAT on irregular treatments in experiment two, viewing the fitness values across generations (not shown) suggests that, were this possible, the increase would have to be dramatic.

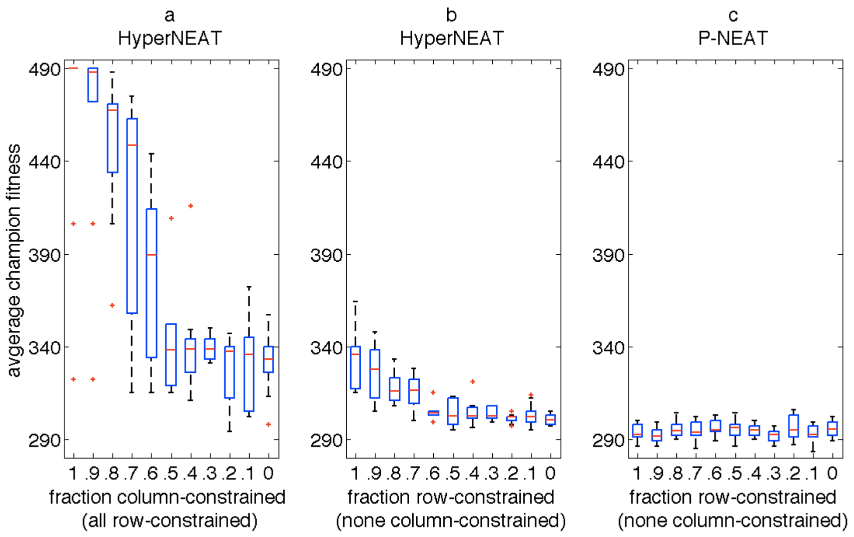


Fig. 1. HyperNEAT and P-NEAT on the Bit Mirroring problem as regularity decreases. **a**) HyperNEAT's performance in experiment one, where within-column constraints are relaxed but within-row constraints remain. b) HyperNEAT's performance in experiment two, where within-column constraints are eliminated and within-row constraints are relaxed. c) P-NEAT's performance in experiment two, which is statistically indistinguishable from its performance on experiment one.

A third experiment continues the comparison of Hyper-NEAT to P-NEAT on problems of decreasing regularity. For this experiment trials lasted 2000 generations, as before, but we conducted 40 trials per treatment due to the high variance between trials. All targets in experiment three are random with respect to row and column, leaving only the regularity inherent in the Bit Mirroring problem. Since this inherent regularity stems from a motif that needs to be repeated for each input node ('zero out links to all outputs but one'), the number of times this motif needs to be repeated decreases with the grid size. Unfortunately, there is no way to decrease this inherent regularity without also decreasing the problem complexity (i.e. the number of weights the algorithm is optimizing). Based solely on problem regularity, P-NEAT should perform better in comparison to HyperNEAT as this type of regularity is decreased. Fig 2. reveals that this is the case. The performance of HyperNEAT degrades to and then falls below that of P-NEAT as the grid size decreases. The overall decline is significant (p<.05 comparing the ratios on the 3x3 treatments vs. those 6x6 and greater). It is not clear why the trend is reversed on the smallest grid size. Note that experiments one and two occurred on a 7x7 grid where the level of inherent regularity provided HyperNEAT an advantage over P-NEAT, even without within-column or within-row regularity, which explains HyperNEAT's superiority for those treatments reported above.

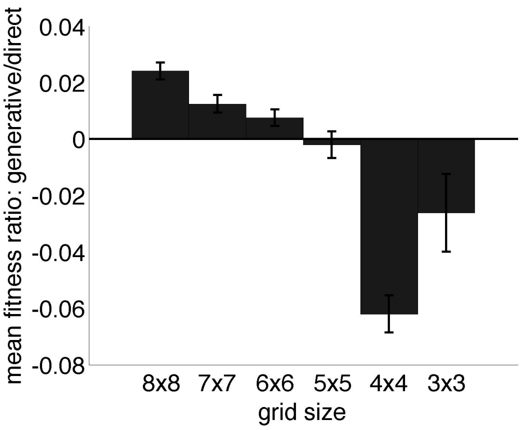


Fig. 2. Comparison of HyperNEAT to P-NEAT as the inherent regularity of the Bit Mirroring problem is decreased, which is accomplished by reducing grid size. Error bars show one standard error of the mean. Ratios are used instead of absolute differences because the allowable fitness ranges change with grid size.

Experiment three shows that once problems are sufficiently irregular and simple, the direct encoding P-NEAT can outperform the generative encoding HyperNEAT. The likely explanation is that HyperNEAT is biased towards creating modular phenotypes and has trouble when the problem features mostly exceptions and little rule. However, even the 3x3 version of the Bit Mirroring problem has some inherent regularity left over. This paper next compares HyperNEAT to P-NEAT on a problem that can be scaled to complete irregularity and where problem complexity remains constant. While there may always be regularities of which an experimenter is not aware, it seems that a completely irregular problem can be created if each link in the neural network phenotype has its own randomly chosen target value. In this 'Target Weights' problem, fitness measures how well the phenotype matches a pre-selected phenotype instead of evaluating a phenotype on a problem with inputs and outputs. A regular version of this problem can be constructed if all target weights are the same randomly chosen value. The regularity can be decreased by lowering the percent of weights that have a repeated target. For the treatment where 50% of the weights are repeated, for example, a 'repeated value' is randomly chosen and that value becomes the target weight for a randomly selected 50% of links. The remaining 50% of links each have a random target chosen independently. This experiment ran faster, allowing 10 trials per treatment of 5000 generations on a 3x3 grid.

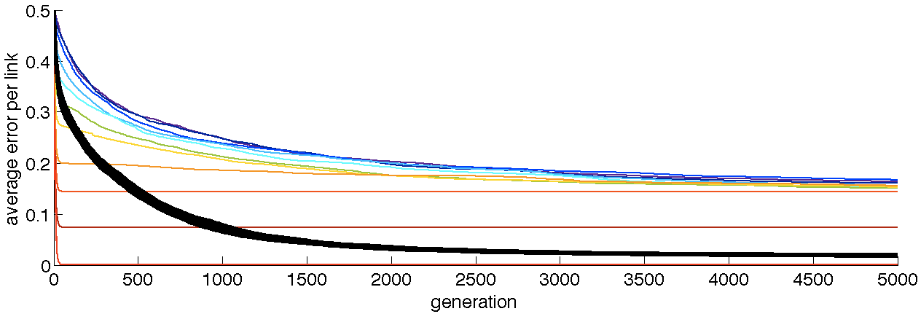


Fig. 3. Comparison of HyperNEAT to P-NEAT on 11 treatments of the Target Weights problem where the percent of targets repeated decreased by 10% from 100 to 0 percent. The initial average error of HyperNEAT (thin lines) is inversely related to the regularity of the treatment (e.g. the lowest line, with almost zero error throughout, is the most regular HyperNEAT treatment, and the highest line is the most irregular HyperNEAT treatment), although HyperNEAT's performance on less regular treatments becomes conflated over time. The performance of P-NEAT (thick lines) was not affected by the regularity of the problem, which is why the lines are overlaid and indistinguishable.

Fig 3 shows that HyperNEAT exploits the regularity of the Target Weights problem early on, but P-NEAT closes the gap fast and eventually outperforms HyperNEAT on all but the most regular treatment. This experiment provides another kind of example where a direct encoding does better compared to a generative encoding as the problem regularity decreases. This experiment also serves as a further illustration both that the performance of HyperNEAT decreases with the regularity of the problem, and that the fitness values, at least at the end of the run, are statistically indistinguishable once the regularity of the problem falls below a relatively high threshold (p>.05 comparing the 0% repeat treatment to all but the 90 and 100 percent repeat treatments). However, the difference in HyperNEAT's performance between treatments early on complicates the story of how HyperNEAT's performance flatlines below a certain regularity threshold, by making it depend on time. Fitness plots across generations from experiments one and two (not shown) do not tell a similar story; in those experiments the less regular treatments have similar fitness scores throughout. A further point of interest is the lack of progress HyperNEAT makes on the highly regular treatments (e.g. where 80 or 90 percent of the targets are repeated). While it exploits the regularity early on, HyperNEAT seems unable to make exceptions to the rule in order to encode the non-conforming link values, as evidenced by the lack of fitness improvements after the initial surge. Unsurprisingly, the P-NEAT trials from this experiment are statistically indistinguishable (p>.05).

Each of the previous experiments have shown how HyperNEAT and P-NEAT perform as a *single type* of regularity is scaled from high to low. Fig. 4 shows how HyperNEAT and P-NEAT perform as the number of *concurrent types* of regularity is decreased. It samples from the first four experiments. While it could have been the case that exploiting one type of regularity prevented the exploitation of others, Fig. 4

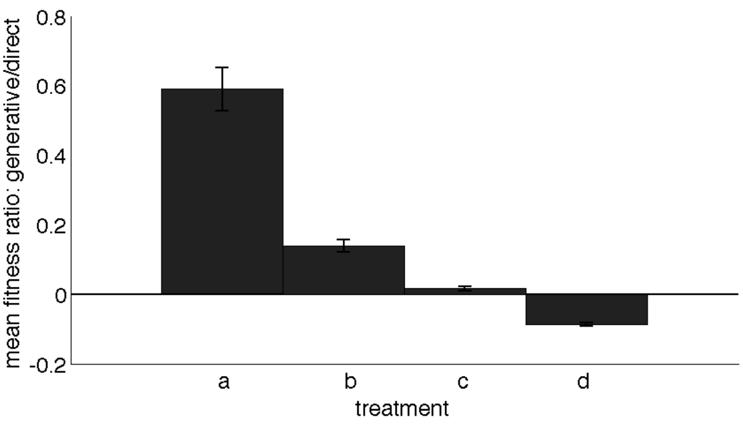


Fig. 4. Comparison of HyperNEAT to P-NEAT across experiments as regularity is decreased. **a**) All targets are constrained to be within the same column and row as their source on the 7x7 Bit Mirroring problem (three types of concurrent regularity). **b**) Targets are only constrained to be in the same row on the 7x7 Bit Mirroring problem (two types of concurrent regularity) **c**) Targets are randomly chosen, leaving only the inherent regularity of the 7x7 Bit Mirroring problem (one type of regularity) **d**) Randomly chosen (no repeated) values on the Target Weights problem (no types of regularity).

reveals that it is possible for HyperNEAT to simultaneously exploit multiple types of regularity. It also demonstrates that the performance of HyperNEAT degrades to, then falls below, that of P-NEAT as concurrent problem-regularity decreases.

4 Discussion, Future Work and Conclusion

The experiments in this paper, which cover four types of regularity from two different problems, paint a consistent picture despite some idiosyncrasies. In general, the HyperNEAT generative encoding showed some difficulty in making exceptions to the rules it discovered. Its performance decreased as problem regularity decreased. Nevertheless, the generative encoding did provide a fitness boost over its direct encoding counterpart on regular problems. The generative encoding's ability to simultaneously exploit concurrent types of regularities meant that the more types of regularity, the larger the boost. However, the generative encoding could only exploit a type of regularity when the amount of regularity within that type was relatively high. This result is not obvious from theoretical considerations and, to the authors' knowledge, has not been reported before. Future work is needed to see if the conclusions drawn from this generative encoding on these two problems apply to most generative encodings on many problems. It would also be interesting to test less extreme types of irregularity. Instead of non-constrained nodes being randomized, for example, they could be offset by a fixed amount. This would still test whether exceptions to the rule can be made, but would test a different, more regular, type of exception. Often the exceptions that need to be made to a rule do not involve radical departures from that rule, and generative encodings may do better at accommodating more subtle variations than those tested here. Finally, while the direct encoding outperformed the generative encoding on irregular problems, as might be expected from the No Free Lunch theorem [12], it was not until the problem was relatively irregular that this transition occurred. P-NEAT only excelled on very small versions of the Bit Mirroring problem and the Target weights problem, where concurrent regularities were few. In fact, it was challenging for the authors to come up with problems irregular enough to provide an advantage to the direct encoding. Even the 7x7 Bit Mirroring problem, which is simple compared to real-world problems, had multiple regularities that could be exploited. It is likely that on most difficult real-world problems, the existence of many types of regularities will provide an advantage to generative encodings. One interesting question this paper raises, however, is whether the level of regularity within each type will be sufficient for a generative encoding to be able to exploit it.

References

1. Hornby, G.S., Pollack, J.B.: Creating High-Level Components with a Generative Representation for Body-Brain Evolution. Artificial Life 8(3), 223–246 (2002)
2. D'Ambrosio, D.B., Stanley, K.O.: A novel generative encoding for exploiting neural network sensor and output geometry. In: Whitley, D., Goldber, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.-G. (eds.) GECCO 2007, pp. 974–981. ACM Press, New York (2007)
3. Gauci, J.J., Stanley, K.O.: Generating Large-Scale Neural Networks Through Discovering Geometric Regularities. In: Whitley, D., Goldber, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.-G. (eds.) GECCO 2007, pp. 997–1004. ACM Press, New York (2007)
4. Gruau, F.: Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process. International Workshop on Combinations of Genetic Algorithms and Neural Networks 6, 55–74 (1992)
5. Gruau, F., Whitley, D., Pyeatt, L.: A Comparison Between Cellular Encoding and Direct Encoding for Genetic Neural Networks. In: Proc. 1st Ann. Conf. on Genetic Programming 1996, pp. 81–89. MIT Press, Cambridge (1996)
6. Stanley, K.O., Miikkulainen, R.: A Taxonomy for Artificial Embryogeny. Artificial Life 9(2), 93–130 (2003)
7. Reisinger, J., Miikkulainen, R.: Acquiring Evolvability Through Adaptive Representations. In: Whitley, D., Goldber, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.-G. (eds.) GECCO 2007, pp. 1045–1052. ACM Press, New York (2007)
8. Nolfi, S., Miglino, O., Parisi, D.: Phenotypic Plasticity in Evolving Neural Networks. In: Proc. Intl. Conf. from Perception to Action. IEEE Press, Los Alamitos (1994)
9. Stanley, K.O.: Compositional Pattern Producing Networks: A Novel Abstraction of Development. Genetic Programming and Evolvable Machines Special Issue on Developmental Systems 8(2), 131–162 (2007)
10. Southan, C.: Has the Yo-Yo Stopped? An Assessment of Human Protein-Coding Gene Number. Proteomics 4(6), 1712–1726 (2004)
11. Stanley, K.O., Miikkulainen, R.: Evolving Neural Networks Through Augmenting Topologies. Evolutionary Computation 10(2), 99–127 (2002)
12. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation 1, 67–82 (1997)

Sub-tree Swapping Crossover, Allele Diffusion and GP Convergence

Stephen Dignum and Riccardo Poli

Department of Computing and Electronic Systems,
University of Essex,
Wivenhoe Park, Colchester, CO4 3SQ, UK
{sandig,rpoli}@essex.ac.uk

Abstract. We provide strong evidence that sub-tree swapping crossover when applied to tree-based representations will cause alleles (node labels) to diffuse within length classes. For a-ary trees we provide further confirmation that all programs are equally likely to be sampled within any length class when sub-tree swapping crossover is applied in the absence of selection and mutation. Therefore, we propose that this form of search is unbiased - within length classes - for a-ary trees. Unexpectedly, however, for mixed-arity trees this is not found and a more complicated form of search is taking place where certain tree shapes, hence programs, are more likely to be sampled than others within each class. We examine the reasons for such shape bias in mixed arity representations and provide the practitioner with a thorough examination of sub-tree swapping crossover bias. The results of this, when combined with crossover length bias research, explain Genetic Programming's lack of structural convergence during later stages of an experimental run. Several operators are discussed where a broader form of convergence may be detected in a similar way to that found in Genetic Algorithm experimentation.

Keywords: Genetic Programming, Search, Crossover Bias, Allele Diffusion, Convergence.

1 Introduction

An intrinsic feature of traditional Genetic Programming (GP) is its variable-size tree-based representation [6,8]. Sub-tree swapping crossover has also, from the inception of GP, been the predominant genetic operator [4,5]. It is essential, therefore, for GP practitioners to understand the biases inherent in using this form of representation and the primary variation operator applied to it.

Recent research has shown that sub-tree swapping crossover will sample exponentially more shorter programs for a-ary trees[1] when applied to a flat fitness landscape in the absence of mutation [7], i.e., when its bias is isolated. This was extended by generalisation to mixed-arity trees in [2] and to true length-classes

[1] Representations made up of internal nodes that have a single common arity, e.g., 2 for the case of Boolean induction problems which use the functions AND, OR, etc.

(from internal node counts) in [3]. Strong empirical support has been found for each generalisation.

One can divide the space of all possible programs into subsets. As we have discussed, one way is to group programs by the number of nodes in the tree representing them. We will call each such set a *length class*. A finer classification would be to divide the programs by their tree shape. This is what we will call a *shape class*. Each program shape is characterised by the number of primitives/nodes of each arity it contains. This can provide a (non-unique) signature for the shape, which we will call an *arity histogram*. Of course, all shapes with a particular arity histogram also have an identical number of nodes. So, if we group programs by their arity histograms we obtain a sub-division of the program space which is between the length class and the program shape in that many shapes (but only one program size) can correspond to an arity histogram.[2]

An assumption (indirectly corroborated numerically) of the original hypothesis in [7] was that all tree shapes within a particular length class for a-ary trees would be equally likely, as all correlations present within the shapes would be removed by the crossover operator. This implies a diffusive process where any node is equally likely to be in any position within the tree shape. If this diffusion process occurs we can assume that sub-tree swapping crossover is unbiased in its exploration of the search space within each length class, i.e., it will explore all programs with equal probability within each length.

The appropriateness of bias (or lack of) is problem dependent (see No Free Lunch Theorems [11]). However, characterising the bias allows us to understand why GP has been successful in solving certain problems or classes of problems. Understanding such bias also allows us to explain how GP searches when areas of neutrality are reached or when selection reduces fitness variance in the population during the later stages of a GP run. It also provides a starting point in the analysis of the effects of combinations of GP operators.

Within Section 2 we briefly explain current findings for length bias. In Section 3 we use a cartesian node reference system to identify all possible positions within a tree. From this we provide evidence of a diffusion process showing that all correlations between nodes are broken by repeated application of sub-tree swapping crossover in the absence of selection and other reproduction operators.

We turn our attention to unique shapes within length classes in Section 4. As predicted, shape classes are shown to have equal occurrence within each length class for a-ary trees, although as predicted in [7] shapes within smaller lengths are more widely sampled than those of larger lengths. Shapes within length classes for mixed-arity trees, however, are not sampled equally. We find that only those within each distinct arity histogram class are sampled in such a way. This extends current research showing us that the repeated application of crossover distributes trees according to their arity histogram. Earlier results for a-ary representations are a special case of this more general result.

[2] Naturally, the distinction between length-class and arity histogram disappears for a-ary trees. Also, in both the single and the mixed-arity cases, the number of terminals is always determined by the rest of the arity histogram.

From our characterisation of crossover's biases we are in a position to explain the lack of structural convergence of GP solutions during experimentation [1, page 278]. Structural convergence is an effect seen in other forms of evolutionary search, notably, Genetic Algorithms (GAs) where it is often used as a stopping criterion for runs. This is discussed in Section 5 along with potential broader convergence detection measures, while in Section 6 we summarise our findings.

2 Length Distributions

In [7] we provided a mathematical model with strong experimental evidence showing that the repeated application of standard sub-tree swapping crossover with uniform selection of crossover points will push a population of a-ary trees towards a limiting distribution of tree sizes called a *Lagrange distribution of the second kind.* This distribution shows a strong tendency to sample programs of small sizes, programs including only one terminal being sampled most often.[3]

This result was generalised in [2] to show that a similar distribution exists for mixed arity trees. As an illustration, Figure 1 shows a theoretical distribution with empirical verification for a population with a mix of internal nodes with arities of 2, 2, and 3, i.e., for that of the Artificial Ant problem [4].

The predictive model used to produce the distribution in Figure 1 is

$$\Pr_g\{n\} = (1 - \bar{a}p_{\bar{a}})\frac{\Gamma(\bar{a}n + 2)}{\Gamma((\bar{a}-1)n + 2)\Gamma(n+1)}(1 - p_{\bar{a}})^{(\bar{a}-1)n+1}p_{\bar{a}}^n \tag{1}$$

where $\Pr_g\{n\}$ is the probability of selecting an individual with n internal nodes, $\Gamma()$ is the Gamma function, $\bar{a}$ is the average of arities in the initial population before crossover is applied, μ_0 is the initial mean tree size within that population, and $p_{\bar{a}}$ is used to simplify the formula and is defined as follows

$$p_{\bar{a}} = \frac{2\mu_0 + (\bar{a} - 1) - \sqrt{((1 - \bar{a}) - 2\mu_0)^2 + 4(1 - \mu_0^2)}}{2\bar{a}(1 + \mu_0)} \tag{2}$$

For $\bar{a} > 1$ the function in Equation (1) is decreasing. It was shown in [2] that increasing the initial mean program size reduces its slope, hence, reducing the bias to sample smaller programs.

Finally, the distribution was generalised once more in [3] to provide predictions based on exact lengths rather than internal nodes. While for a-ary trees there is a one-to-one mapping between length and internal nodes, for mixed arity trees there are occasional, if minor, discrepancies at shorter lengths and the generalisation is approximate. Nonetheless, the match between the model and experimental results is very good, any discrepancies disappearing as program size increases. The reasons for the minor deviations at shorter lengths are explained in the following sections.

[3] The 90/10 node-selection policy commonly used in GP to counter this effect was also shown in [2] to have little effect on the sampling of all but the smallest classes.

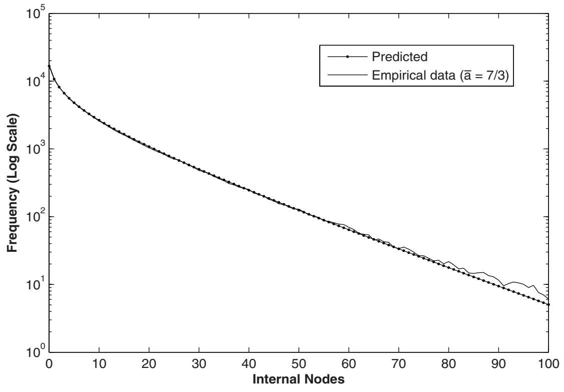


Fig. 1. Comparison between empirical and predicted internal node distributions, in the absence of selection and mutation, for trees made up of a mix of 2, 2, and 3 arity functions ($\bar{a}$=7/3) initialised with FULL method (depth = 3, initial mean size $\mu_0 = 21.48$, mean size after 500 generations $\mu_{500} = 23.51$). Population Size 100,000 individuals, empirical results averaged over 20 runs.

3 Allele Diffusion

Our first task is to test the assumption that crossover will remove any correlations between nodes ensuring that all node labels are equally likely to be found at any position within trees created purely from the application of crossover.

Earlier work provided theoretical and empirical evidence to support this claim for linear GP [10], where only internal nodes of arity 1 were used. This, of course, is a specific case of the a-ary assertion in [7].

We have chosen to implement the technique used in [10] where a node marker or 'dye' is applied at specific positions within trees during initialisation. The amount of dye is then recorded for each node position in subsequent generations.

With linear GP it is possible to compare directly node positions within length classes. This is not true, however, for a-ary trees or those with mixed arities. We have chosen, therefore, to implement a cartesian node reference system to assign unique node positions for all possible trees based upon the maximum arity that may be used. The exact method is described in [9]. However, it can simply be described as producing a template based on a maximal tree, i.e., one where only the largest arity is used without terminals up until a maximum depth. Each node is assigned a unique integer number in the order of left-to-right breadth-first traversal, 1 being the position of the root node.

For each set of experiments a population of 100,000 individuals was used. Dye was placed either at reference 1 (the root node) or at reference 5. These positions have been chosen carefully to ensure dye was applied once to every tree during initialisation for all of our arity mixes, hence, simplifying theoretical calculations. For all experimentation a flat fitness landscape was used and sub-tree swapping crossover with uniform selection of crossover points was applied

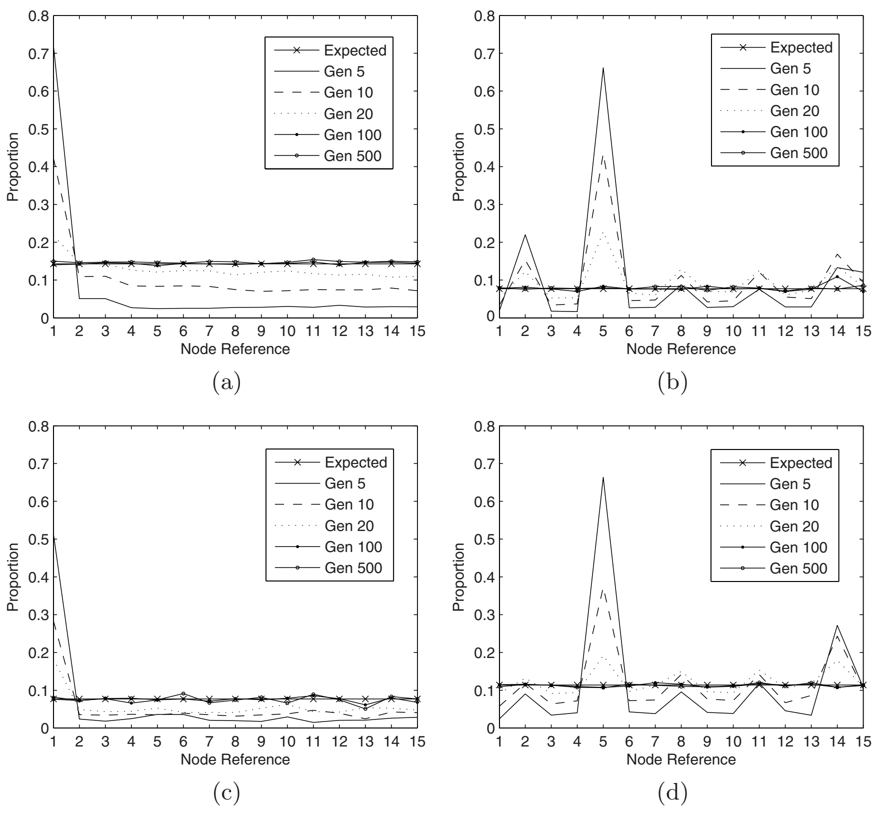


Fig. 2. Plots of the relative proportion of non-terminal dye alleles vs node references for: (a) 2-ary programs of length 11, initial dye reference 1, (b) 3-ary programs of length 13, initial dye reference 5, (c) mixed arity 1, 2, 3, 4 & 5 programs of length 11, initial dye reference 1, (d) mixed arity 2, 2 & 3 programs of length 13, inital dye reference 5. Note, selected tree lengths are smaller than the smallest trees created by the initialisation method hence data are not recorded for generation 0.

with no mutation or reproduction. All programs were initialised using the FULL method with a depth of 3 (depth 0 being the root node) and all results have been averaged over 20 independent runs.

In Figure 2a we can see that for the proportion of internal nodes with dye, for 2-ary trees of length 11, we move rapidly to our expected value at each of the first fifteen possible node references.[4]

For 2-ary trees initialised with the FULL method with depth 3, each tree will have only one dye node for each of the possible seven internal nodes, hence,

[4] Note, it is possible for internal nodes to reach a position of 31 using our reference system for 2-ary trees of length 11. A limit of 15 was chosen for consistency across experimentation.

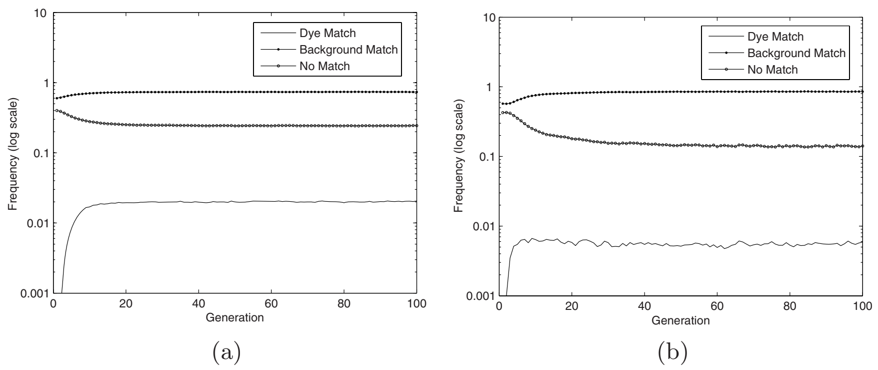


Fig. 3. Plots of the mean relative frequency of co-occurrence of pairs of non-terminal alleles vs. generation for 2-ary (a) and mixed arity 1, 2, 3 , 4 & 5 (b) programs of length 11. Population initialised as Figure 2.

after diffusion has taken place we expect all positions to have a dye proportion of 1/7 for internal nodes. Consistently similar results, i.e., convergence to predetermined predicted proportions are seen in additional experiments for 3-ary trees, and mixed arity trees of 1, 2, 3, 4 & 5 and 2, 2 & 3 arity nodes.[5] These are shown in Figures 2b-d respectively.

We next turn our attention to co-occurrence of pairs of non-terminals, i.e., whether we can consistently see any correlation between dye positions. In order to do this for each generation a 15 by 15 matrix is produced. Each row and column records the first 15 positions within the node reference system. For the first row we determine if the first node is dye or background then for each column we then determine whether this matches for any of the other positions and record the match, or lack of, in the corresponding position in our matrix, i.e., row determined by node under investigation, column for nodes to be matched. Diagonals in the matrix are ignored as we will always obtain a match. In Figure 3a we can see that for 2-ary trees initialised with dye at the root position we quickly move to values predicted by a diffusive process. Dye sits on the diagonal for the initial generation and hence is not recorded but then we apply crossover and after approximately 20 generations we have reached our theoretical proportions: $(1/7)^2 \approx 0.020408$ for dye matching, $(6/7)^2 \approx 0.73469$ for background matching, and $2(1/7)(6/7) \approx 0.24490$ for no match. The same is true for our mixed arity trees. For example in Figure 3b our population of 100,000 individuals was initialised with an average of 1,297,856.85 internal nodes, 100,000 of which where marked with dye, our theoretical value for dye co-occurrence is $(100,000/1,297,856.85)^2 \approx (0.07705)^2 \approx 0.00594$. Background matching is, therefore, $(1 - 0.07705)^2 \approx (0.92295)^2 \approx 0.85184$ and finally our no match value will be $2(0.07705)(0.92295) \approx 0.14223$.

[5] All experimentation shown was subjected to a $\chi^2_{10\%}$ test which showed support for the assertion that the first 15 positions, at generation 100, would each contain a number of nodes determined by initial population proportions.

Each of these values is also obtained within 10 to 20 generations. Similar results were also found for our 3-ary and 2, 2 & 3 mixed arity experiments.[6] See [10] for similar results for linear GP, i.e., 1-ary trees.

4 Shape Bias

There is one final aspect of sub-tree swapping crossover that we can analyse before we complete our picture: how we sample shapes within length classes. The length distribution described in [7] is derived from an expectation that all shapes will be sampled uniformly within length classes for a-ary trees. In Figure 4, we can indeed provide experimental evidence for 2-ary trees for our length classes chosen. However, looking at mixed arities we can see that there is a distinct bias to sample certain shape classes within each length. It was found, however, (see Table 1 as an example) that shapes with same arity histogram would be sampled uniformly. This shape bias for mixed arities is easily explained if we look at the dynamics of the proportion of primitives of each arity in the population. On average this form of crossover will replace as much as it removes; this also holds true for node arities. To illustrate, see Figure 5 as an example of how the proportion of primitives of each arity stays constant in a population when sub-tree swapping crossover only is applied for our mixed arity experiments described earlier. There is, therefore, no bias to remove or resample certain higher or lower arities. So, not only does average size remain constant under repeated application of crossover, but also the proportions of each arity will remain constant within the population. Therefore, any (note, highly sampled) smaller shapes without

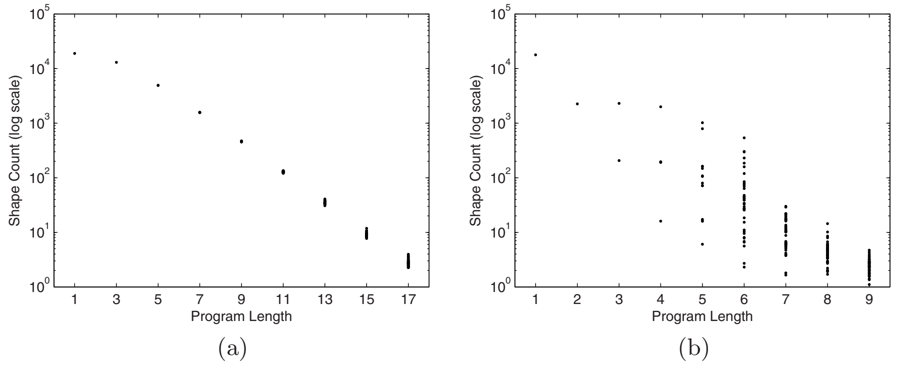


Fig. 4. Scatter plots of shape counts for 2-ary (a) and mixed arity 1, 2, 3, 4 & 5 (b) programs, first 9 possible lengths at generation 500. Population initialised as Figure 2. Note, there are far more possible shapes for larger length classes. Also, these classes are sampled far less often than those of smaller lengths.

[6] For all experiments tree lengths up to a maximum of 40 nodes were analysed, each showed similar results.

Table 1. Averaged counts at generation 500 for all program shapes for 2, 2 & 3 arity programs of length 7. Population initialised as in Figure 2.

S-Expression	Count
(2 0 (2 0 (2 0 0)))	407.10
(2 0 (2 (2 0 0) 0))	407.95
(2 (2 0 0) (2 0 0))	401.05
(2 (2 0 (2 0 0) 0))	404.25
(2 (2 (2 0 0) 0) 0)	410.40
(3 0 0 (3 0 0 0))	258.75
(3 0 (3 0 0 0) 0)	258.05
(3 (3 0 0 0) 0 0)	258.75

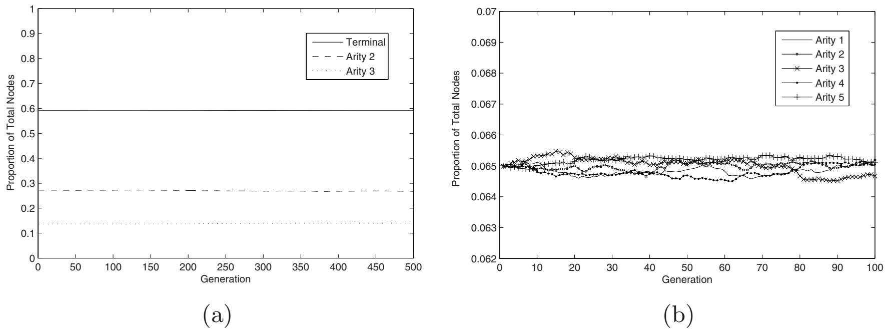


Fig. 5. Plots of the proportions of arities for each generation. (a) shows the first 500 generations for a population initialised with 2, 2 & 3 arities. (b) shows the first 100 generations of a population initialised with 1, 2, 3, 4 & 5 arities, note the highly reduced scale in this example. Due to the reduced scaling terminals are not shown in (b) but follow a consistent proportion as shown in (a) in this case centering tightly around a proportion of 0.675. Populations initialised as in Figure 2.

an equal proportion of arities, or those that can be produced using only a single arity, will reduce those node arities available for larger classes. Further work is required to produce a model to exactly predict such proportions. However, we do know that the generalised model for mixed arities (Equation 1) has been corroborated by extensive empirical work, so such a model must explain why such a generalisation has been successful.

5 Convergence

First suggested in [10], we can now provide strong evidence that GP's inability to structurally converge is caused primarily through crossover's bias to first distribute a population in terms of length and arity histogram and then to diffuse node labels within those classes. As fitness converges during the later stages of a run, crossover sampling will become predominant. Hence, the processes described in this paper will prevent any structural convergence taking place. No matter how strong the selection scheme, e.g. even if the mating pool was populated solely by copies of a single individual (say by using a tournament size equal to that of the population), the resulting child population created by sub-tree swapping crossover would first contain individuals of differing lengths and secondly node labels would be dispersed within those individuals.[7] This would not be true in a GA system using n-point crossover acting on traditional fixed length vector representations as there is no opportunity to alter individual lengths or to move node labels to different locations.

Although GP using sub-tree swapping crossover will prevent convergence to a single syntactic structure, it will start to search within ever tighter bounds and begin to resample heavily smaller classes (see [3] for details). With this in mind we can suggest possible run stopping criteria based solely on convergence as found in GAs. A very simple method would be to determine the undue influence of crossover by detecting a greater ratio of smaller programs. An inexpensive resampling measure based on simple program hashes could also be used, possibly causing run termination when a program has been resampled a pre-specified number of times. Additional more sophisticated methods may look at the length distribution as a whole, i.e., a convergence to the theoretical distribution or in conjunction with fitness measures such as a corresponding reduction in fitness variance.

6 Conclusions and Future Work

This paper has analysed the biases presented by GP sub-tree swapping crossover. We have provided strong evidence that there is a diffusive process that takes place within length classes when sub-tree swapping crossover is repeatedly applied to a flat fitness landscape in the absence of selection. All node labels (alleles) are equally likely to be found within any possible node position for each length class.

We now know that program shapes will be uniformly sampled within arity histogram classes. a-ary trees are a special case in that there is only one arity histogram per length class. Hence, programs will be sampled uniformly within each length. This, however, is not true for mixed arities and a more sophisticated process is taking place. The reasons for this lie within the constant population proportions of each arity during each generation and the highly sampled smaller programs with unequal arity proportions.

In the future we hope to be able to develop a mathematical model similar to that described in [7] to provide the probability of an individual containing a certain proportion of internal node arities. This model will need to explain the theoretical and empirical results found within this paper and those presented in [7,2,3].

Although we now know that GP using sub-tree swapping crossover is highly unlikely to converge in terms of individual program structure, we do have an understanding of a broader, population based, form of structural convergence. This

[7] Barring the unlikely situation where the same crossover points are chosen in all cases.

allows us to propose a set of convergence measures that may be used for stopping conditions similar to those found in GA experimentation. Further research is required to establish the effectiveness of such measures.

References

1. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, San Francisco (1998)
2. Dignum, S., Poli, R.: Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In: Thierens, D., et al. (eds.) GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, London, vol. 2, pp. 1588–1595. ACM Press, New York (2007)
3. Dignum, S., Poli, R.: Crossover, sampling, bloat and the harmful effects of size limits. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 158–169. Springer, Heidelberg (2008)
4. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
5. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge (1994)
6. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2002)
7. Poli, R., Langdon, W.B., Dignum, S.: On the limiting distribution of program sizes in tree-based genetic programming. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 193–204. Springer, Heidelberg (2007)
8. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming (With contributions by J. R. Koza) (2008), `http://lulu.com`, `http://www.gp-field-guide.org.uk`
9. Poli, R., McPhee, N.F.: General schema theory for genetic programming with subtree-swapping crossover: Part I. Evolutionary Computation 11(1), 53–66 (2003)
10. Poli, R., Rowe, J.E., Stephens, C.R., Wright, A.H.: Allele diffusion in linear genetic programming and variable-length genetic algorithms with subtree crossover. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B. (eds.) EuroGP 2002. LNCS, vol. 2278, pp. 212–227. Springer, Heidelberg (2002)
11. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)

How Single Ant ACO Systems Optimize Pseudo-Boolean Functions

Benjamin Doerr, Daniel Johannsen, and Ching Hoo Tang

Max-Planck-Institut für Informatik
Saarbrücken, Germany

Abstract. We undertake a rigorous experimental analysis of the optimization behavior of the two most studied single ant ACO systems on several pseudo-boolean functions. By tracking the behavior of the underlying random processes rather than just regarding the resulting optimization time, we gain additional insight into these systems. A main finding is that in those cases where the single ant ACO system performs well, it basically simulates the much simpler (1+1) evolutionary algorithm.

1 Introduction

In 1991, Dorigo, Maniezzo and Colorni [4] introduced the concept of Ant Colony Optimization (ACO). Since then ACO algorithms have been applied successfully to many kinds of combinatorial problems, e.g., the famous Travelling Salesman Problem. See the book by Dorigo and Stützle [5] and the references therein.

In the last few years, theoretical research has been started to gain an understanding of why these methods are so successful. Since the probability space describing a run of a typical ACO system is extremely complicated, theoretical works concentrated on the runtime behavior of two ACO systems involving a single ant only, namely 1–Ant and the Max–Min Ant System (MMAS).

The algorithm 1–Ant was proposed by Neumann and Witt [11]. It is an adaption of the more general Graph–Based Ant System introduced by Gutjahr [7] to allow optimizing (non-graph based) pseudo-boolean functions. Gutjahr and Sebastiani [8] gave the first rigorous runtime analysis of the MMAS and showed that on certain needle-in-a-haystack functions their ACO system beats the classical (1+1) evolutionary algorithm ((1+1) EA) (for both algorithms a version was used that does not accept new solutions of equal fitness). Neumann and Witt [11] conducted the runtime analysis of 1–Ant on the pseudo-boolean function OneMax (counting the number of ones in an n-bit string). While obviously a highly simplified problem, the analysis was far from simple. The main outcome of the analysis is that the optimization time depends crucially on the major parameter, the so-called evaporation factor ρ. Here, crucially means that there is a relatively sharp distinction between quite efficient optimization and exponential run-time behavior. Additionally, it was observed that for ρ very close to one, 1–Ant exactly simulates the (1+1) EA, which was rigorously analyzed in [6]. In [2] the pheromone model used in [11] was replaced by a simpler, but equivalent model, in which the pheromone values equal the probabilities of ants walking along a particular edge in the construction graph. In [3] an analysis of 1–Ant was carried out for the functions LeadingOnes and BinaryValue. As in the OneMax case, a phase transition from exponential to polynomial runtime with a threshold evaporation value could be observed.

The algorithm MMAS by Stützle and Hoos [12] was studied by Gutjahr and Sebastiani [8] and Neumann, Sudholt and Witt [9]. Roughly speaking, these works show that several variants of the MMAS less critically depend on the choice of the parameter ρ in the sense that there is no sharp phase transition between polynomial and exponential runtime as observed with 1–Ant. More recently, the benefits and shortcomings of hybridizations of the MMAS with local search strategies have been investigated [10].

Some of the run-time analyses sketched above could be read as that single ant ACO system are competitive approaches to optimize pseudo-boolean functions. To further study this aspect, we conduct a rigorous experimental analysis of these two single ant ACO systems on several pseudo-boolean fitness functions (OneMax, LeadingOnes, and linear functions with random weights). To gain an understanding how these algorithms work, we track a number of theory–guided indicators (other than the resulting optimization time) during the runs of 1–Ant [11] and the MMAS [12]. For both algorithms we use the pheromone system described in [2] which is equivalent to the one used in [8], [11], and [3].

Our main finding is that whenever one of the two ACO systems for a certain choice of ρ has an at least roughly reasonable run-time, then its optimization behavior is very similar to that of the (1+1) EA. This shows that the pessimistic assumptions repeatedly used in the proofs of the results mentioned above are real, and in consequence, indicates that the upper bounds on the optimization time proven there probably cannot be improved. Our analysis of the optimization behavior fits well to the fact that we rarely observe that one of the two single ant ACO systems finds the optimum significantly faster than the (1+1) EA [1].

2 Single Ant ACO and the (1+1) EA

Given a *fitness function* $f\colon \{0,1\}^n \to \mathbb{R}$ on the bit-strings of length n, a *single ant ACO algorithm* successively generates candidate solutions $S^{(t)} \in \{0,1\}^n$ according to the *pheromone values* $p^{(t)} \in [0,1]^n$. We understand this sampling process as a random walk of a single ant on the directed *construction graph* depicted in Figure 1. At each vertex v_{i-1} the ant chooses one of the two outgoing

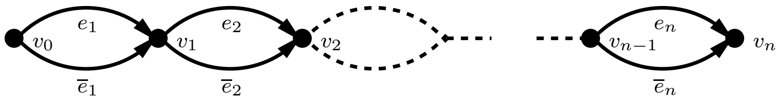


Fig. 1. Bit-strings are represented by ant walks on the simplified chain graph

edges e_i or $\overline{e}_i$ with probability equal to the *pheromone value* $p_i^{(t)}$ or $1 - p_i^{(t)}$, respectively. If the ant chooses e_i, we have $S_i^{(t)} = 1$ and $S_i^{(t)} = 0$ otherwise.

```
**AntWalk**($p$)
1  **for** $i \in \{1, \dots, n\}$ **do** choose $S_i \in \{0,1\}$ with $\Pr(S_i = 1) = p_i$
2  **return** $S$
```

Initially, all pheromone values are 1/2. Hence, the first ant performs a true random walk. Later, updates to the pheromone values are triggered by certain ant walks. In this case, a certain amount of pheromone evaporates from all edges and then the pheromone values of the edges the ant traverses are reinforced. The amount of both, evaporation and reinforcement, is governed by the algorithm's main parameter, the *evaporation factor* $\rho \in [0, 1]$.

```
**Update**($p, S, \rho$)
1  **for** $i \in \{1, \dots, n\}$ **do**
2     **if** $S_i = 1$ **then** $p'_i := \min\{(1-\rho) \cdot p_i + \rho, 1-\frac{1}{n}\}$ **else** $p'_i := \max\{(1-\rho) \cdot p_i, \frac{1}{n}\}$
3  **endfor**
4  **return** $p'$
```

In this theoretical investigation, we run both algorithms for a number of $t_{\max} \in \mathbb{N}$ generations and then return the best solution found so far. In practice, other stopping criteria might be more appropriate.

```
**1–Ant** ($f, t_{\max}, \rho$)
 1  $p^{(0)} := (1/2, ..., 1/2)$
 2  $S_{\max} :=$ **AntWalk**($p^{(0)}$)
 3  $p^{(1)} :=$ **Update**($p^{(0)}, S_{\max}, \rho$)
 4  **for** $t$ **from** 1 **to** $t_{\max}$ **do**
 5     $S^{(t)} :=$ **AntWalk**($p^{(t)}$)
 6     **if** $f(S^{(t)}) \geq f(S_{\max})$ **then**
 7        $S_{\max} := S^{(t)}$
 8        $p^{(t+1)} :=$ **Update**($p^{(t)}, S_{\max}, \rho$)
 9     **endif**
10  **endfor**
11  **return** $S_{\max}$
```

```
**MMAS** ($f, t_{\max}, \rho$)
 1  $p^{(0)} := (1/2, ..., 1/2)$
 2  $S_{\max} :=$ **AntWalk**($p^{(0)}$)
 3  $p^{(1)} :=$ **Update**($p^{(0)}, S_{\max}, \rho$)
 4  **for** $t$ **from** 1 **to** $t_{\max}$ **do**
 5     $S^{(t)} :=$ **AntWalk**($p^{(t)}$)
 6     **if** $f(S^{(t)}) \geq f(S_{\max})$ **then**
 7        $S_{\max} := S^{(t)}$
 8     **endif**
 9     $p^{(t+1)} :=$ **Update**($p^{(t)}, S_{\max}, \rho$)
10  **endfor**
11  **return** $S_{\max}$
```

1–Ant and the MMAS both simulate one of two well–known randomized search heuristics if the evaporation factor is zero or one. For $\rho = 0$, they simply perform random search, and for $\rho = 1$, they precisely simulate the (1+1) EA with mutation probability $1/n$.

3 The Experimental Setup

The classical mean to measure the performance of a randomized search heuristic is the *optimization time* T, which is the number of fitness evaluations needed to find the optimal solution. For efficiency reasons, we introduce an artificial upper bound of $t_{\max} = 1000000$, i. e., $T = \min\{t \in \mathbb{N} \mid f(S^{(t)})$ is optimal or $t = t_{\max}\}$.

To analyze the optimization behavior, we monitor a number of theory–guided indicators measuring the algorithm's progress at every single step $t \in \mathbb{N}$ of the run. In particular, we investigate the fitness of the current solution $f(S^{(t)})$, its expectation $\mu^{(t)} := \mathrm{E}[f(S^{(t)})]$ and variance $\nu^{(t)} := \mathrm{Var}[f(S^{(t)})]$, the fitness of the best solutions so far $f_{\max}^{(t)} := \max_{r \leq t} f(S^{(r)})$, the number of pheromone values $mm := |\{i \mid p_i^{(t)} \in \{1/n, 1-1/n\}\}|$ attaining one of the boundary values $1/n$ and $1-1/n$, and the probability $P^{(t)} := \Pr(f(S^{(t)}) \geq f_{\max}^{(t)})$ of accepting the current solution.

We also investigate the average values $\overline{P}$, $\overline{\nu}$, and $\overline{mm}$ of $P^{(t)}$, $\nu^{(t)}$, and $mm^{(t)}$ over the interval $[0.25\,T, 0.75\,T]$. This interval was chosen to eliminate possible side-effects at the beginning and the end of a run.

We study the progress behavior of the two ant optimization algorithms 1–Ant and the MMAS on different pseudo-boolean fitness functions $f\colon \{0,1\}^n \to \mathbb{R}$. We regard *random linear functions* $f(S) = \sum_{i=1}^n w_i S_i$, where the weights $w_1, \ldots, w_n$ are chosen independently and uniformly at random in $(0,1]$, and then normalized to add up to n. Clearly, the normalization does not change the behavior of any of the algorithms, but eases comparing the results for different functions. Furthermore, we regard the two classical pseudo-boolean test functions $\textsc{OneMax}(S) = \sum_{i=1}^n S_i$ and $\textsc{LeadingOnes}(S) = \sum_{k=1}^n \prod_{i=1}^k S_i$. Clearly, $S^* = (1, \ldots, 1)$ is the unique maximum of all these functions having fitness $f(S^*) = n$. In the experiments, we use a problem size of $n = 1000$ for OneMax and random linear functions, and a problem size of $n = 200$ for LeadingOnes.

Also, note that for LeadingOnes, $\mu^{(t)}$ and $\nu^{(t)}$ as defined above are heavily influenced by the fact that with probability around $1/e$, one of the leading one–bits (having pheromone value $1 - 1/n$) will be set to zero. Since such a solution will not be accepted anyway, for LeadingOnes we modify the definitions of $\mu^{(t)}$ and $\nu^{(t)}$ to be the expectation and variance conditional on that none of these leading bits is zero.

We omit the details on how to actually compute the progress indicators for these test functions. In all but one case, this can be done efficiently in linear time or via dynamic programming in quadratic time. For arbitrary linear function, however, $P^{(t)}$ cannot be computed efficiently. In consequence, we cannot provide $P^{(t)}$ and $\overline{P}$ for random linear functions.

We perform case studies to analyze the time–dependent indicators $f(S^{(t)})$, $f_{\max}^{(t)}$, $\mu^{(t)}$, $\nu^{(t)}$, $P^{(t)}$, and $mm^{(t)}$. That is, for all algorithms and test functions, we conduct twenty runs each for at least twenty different ρ-values, graphically depict the indicators and single out typical runs for representative values of ρ. In all cases, we see that the indicators for random linear functions and OneMax behave highly similar. For this reason, in the following two sections we present and discuss plots for OneMax only, since here we also have the indicator $P^{(t)}$.

To measure the average indicators $\overline{\nu}$, $\overline{P}$, and $\overline{mm}$, we performed 20 runs for both algorithms on all fitness functions and several values of ρ. We then average the average indicators over all runs. Since the success of 1–Ant depends sharply on ρ, we happen to never average over successful and unsuccessful runs simultaneously. For unsuccessful runs we also record the final values of $f_{\max}$ and

P. For reasons of space, we can only present a tiny fraction of the data collected. Much additional material can be accessed in [1].

4 Experimental Results for 1–Ant

In this section, we present our experimental work concerning the single ant ACO system 1–Ant. In [11] it was shown that the expected optimization time of 1–Ant on OneMax is polynomial if $\rho = 1 - 1/n^{\epsilon}$ with fixed $\epsilon > 0$ and in [2] that for $\rho = o(1/\log n)$ it becomes super–polynomial. On the function LeadingOnes the expected optimization time was shown [3] to be quadratic for constant ρ, polynomial for $\rho = \Omega(1/\log n)$, and again super-polynomial for $\rho = o(1/\log n)$. Note that $\rho = 2n\tilde{\rho}/(1-\tilde{\rho}+2n\tilde{\rho})$ for the evaporation factor $\tilde{\rho}$ in [8], [11] and [3].

Since we argue that for efficient runs of 1–Ant the optimization behavior strongly resembles that of the (1+1) EA, let us first present a typical run for the (1+1) EA. Note that 1–Ant for $\rho = 1$ exactly simulates the (1+1) EA, so we can reuse our test environment here. Figure 2 shows such a typical run as a chart of the indicators $f_{\max}^{(t)}$, $\mu^{(t)}$, $\nu^{(t)}$, $P^{(t)}$, and $mm^{(t)}$. As it is easy to see, the (1+1) EA improves the fitness relatively fast. This is natural, since the probability $P^{(t)}$ of finding an acceptable solution remains very large during the whole run. For the same reason, $f_{\max}^{(t)}$ and $\mu^{(t)}$ are that close together.

We now analyze one representative run of 1–Ant on OneMax for each of the values $\rho = 0.4$, 0.2, 0.1, and 0.05, which give a good overview of the different behaviors of 1–Ant. The plots are depicted in Figure 3.

For $\rho = 0.4$, we easily identify a behavior highly similar to that of the (1+1) EA. In a very short initial phase of approximately 130 iterations, almost all pheromone values are pushed to their extreme values and the variance drops from its initial value of $n/4$ to a value close to one. Note that $f_{\max}$ remains close to $n/2$ in this initial phase. In consequence, this means that half of the

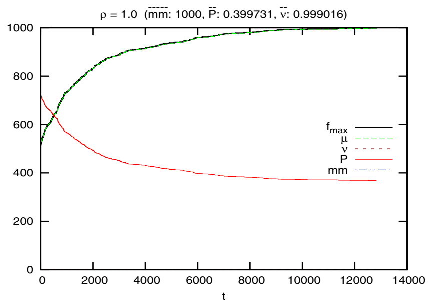


Fig. 2. A typical run of the (1+1) EA on OneMax. Since $\nu^{(t)} = 1-1/n$ and $mm^{(t)} = n$ for all t, these curves coincide with the lower and upper boundaries of the chart. Also, $f_{\max}^{(t)}$ and $\mu^{(t)}$ are too close to each other to be distinguished. In the chart we rescale $p^{(t)}$ from $[0,1]$ to $[0,1000]$.

pheromone values attain the maximum value and half the minimum. By symmetry, they are randomly chosen as is the initial solution in the (1+1) EA. From this point on, the optimization behavior of the 1–Ant closely resembles the one of the (1+1) EA. This is easily seen from the curves and the average values $\overline{P}$, $\overline{mm}$ and $\overline{\nu}$. In the chart it seems that $P^{(t)}$ oscillates heavily, but as the average $\overline{P}$ indicates, these downward dents are only short-term occurrences. They stem mainly from the fact that after an improvement in the fitness, the pheromone values affected take a few iterations until they hit the extreme values again.

The chart for $\rho = 0.2$ still shows many signs of an (1+1) EA-like behavior. However, we also see first short phases of stagnation. At $t = 467$, a solution of value 663 is found (+11 to the previous best). Due to the slower pheromone update with $\rho = 0.2$, this increases the expected value of the next solution only from 642 to 646, and spoils the probability of finding an acceptable solution down to a mere 0.4%. In consequence, it takes the 1–Ant a long 450 iterations to find an as good solution again (at time $t = 917$).

For $\rho = 0.1$, the phenomenon just described becomes much more dominant. We now have several long phases in which no solution is accepted. Finally, for $\rho = 0.05$ this pattern is so strong that no solution is found within one million iterations. With $\overline{P}$ around $2 \cdot 10^{-5}$, it is very hard to find an acceptable solution. Recall that $P^{(t)} = 2 \cdot 10^{-5}$ means that an expected number of 50.000 iterations are necessary to generate a solution that is accepted.

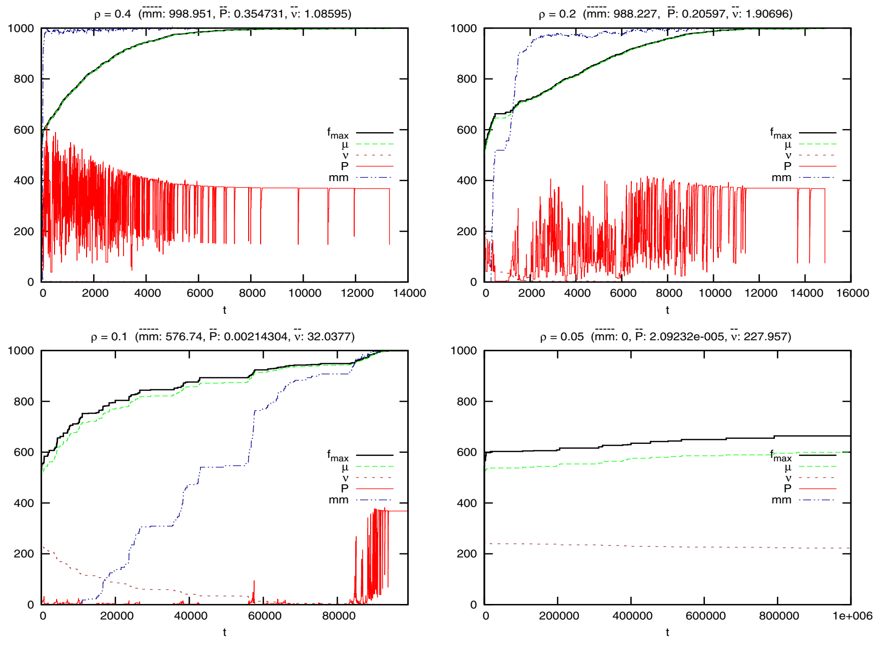


Fig. 3. Four typical runs of 1–Ant on OneMax for $\rho = 0.4$ (top left), 0.2 (top right), 0.1 (bottom left), and 0.05 (bottom right).

What we just extracted from the charts in Figure 3 is also visible from Table 2. For both, random linear functions and OneMax, we observe the expected behavior. For $\rho \geq 0.2$, we have a (1+1) EA-like optimization behavior. All but very few pheromone values are at their extreme values. Those who are not, still are very close to the extreme values, as can be deduced from the variance. For $\rho \leq 0.1$, finding acceptable solutions becomes increasingly hard. $\overline{P}$ values of 10^{-3} or less indicate that fewer than every thousandth solution generated is actually accepted. From $\rho \leq 0.06$ on ($\rho \leq 0.08$ for linear functions), as few solutions are accepted that (a) one million runs never sufficed to find the optimum, (b) the variance is close to the maximum value of 250 (approx. 330 for random linear functions), indicating that most pheromone values are close to their initial values. To add a number, averaging over 20 runs with $\rho = 0.05$ we found that less than 49 (of the one million generated) solutions are accepted.

We see very similar results if we use LeadingOnes as fitness function. For reasons of space, we only present the data in table form (Table 1). Again, we see that for $\rho \geq 0.2$, most pheromone values are at their extreme values and the optimization time is not much different from the case $\rho = 1$, which again is the (1+1) EA. The phase transition happens for slightly smaller ρ values. Up to $\rho = 0.08$, we see still reasonable optimization times. From then on, however, the optimization time increases again drastically and $\overline{P}$ falls to ridiculously small

Table 1. Indicators for the behavior of 1–Ant optimizing LeadingOnes for different values of ρ. For $\rho = 1.0$, 0.5, and 0.1 all runs find the optimum in at most 10^6 steps, for $\rho = 0.01$, 0.005, and 0.001 none. We omit $\rho = 0.05$ to avoid the bias caused by T not reaching 10^6 in all of the runs.

	OneMax						linear functions			
ρ	T	$\overline{mm}$	$\overline{P}$	$\overline{\nu}$	$f_{\max}^{(t_{\max})}$	$P^{(t_{\max})}$	T	$\overline{mm}$	$\overline{\nu}$	$f_{\max}^{(t_{\max})}$
1.0	18246	1000	0.38	0.999			17773	1000	1.332	
0.5	10953	998.8	0.36	1.104			16756	999.6	1.350	
0.1	146895	352.4	$1.82 \cdot 10^{-3}$	66.038			579284	414.6	52.153	
0.05	1000000	0	$1.82 \cdot 10^{-5}$	213.223	691	$7.68 \cdot 10^{-6}$	1000000	0	298.104	673
0.01	1000000	0	$3.96 \cdot 10^{-6}$	248.732	582	$1.95 \cdot 10^{-6}$	1000000	0	332.857	595

Table 2. Results for 1–Ant optimizing the fitness functions OneMax and random linear functions. All numbers are the averages over 20 runs. For $\rho = 0.05$ and $\rho = 0.01$, where no run is successful, we also list the average optimum and acceptance probability at $t_{\max} = 1000000$.

ρ	T	$\overline{mm}$	$\overline{P}$	$\overline{\nu}$	$f_{\max}^{(T)}$	$P^{(T)}$
1.0	34768	200	0.55	0.03		
0.5	33964	197	0.54	0.06		
0.1	35175	177	0.47	0.60		
0.01	1000000	0	$4.47 \cdot 10^{-6}$	3.33	23	$1.90 \cdot 10^{-6}$
0.005	1000000	0	$3.52 \cdot 10^{-6}$	2.49	22	$1.64 \cdot 10^{-6}$
0.001	1000000	0	$3.20 \cdot 10^{-6}$	2.09	21	$1.09 \cdot 10^{-6}$

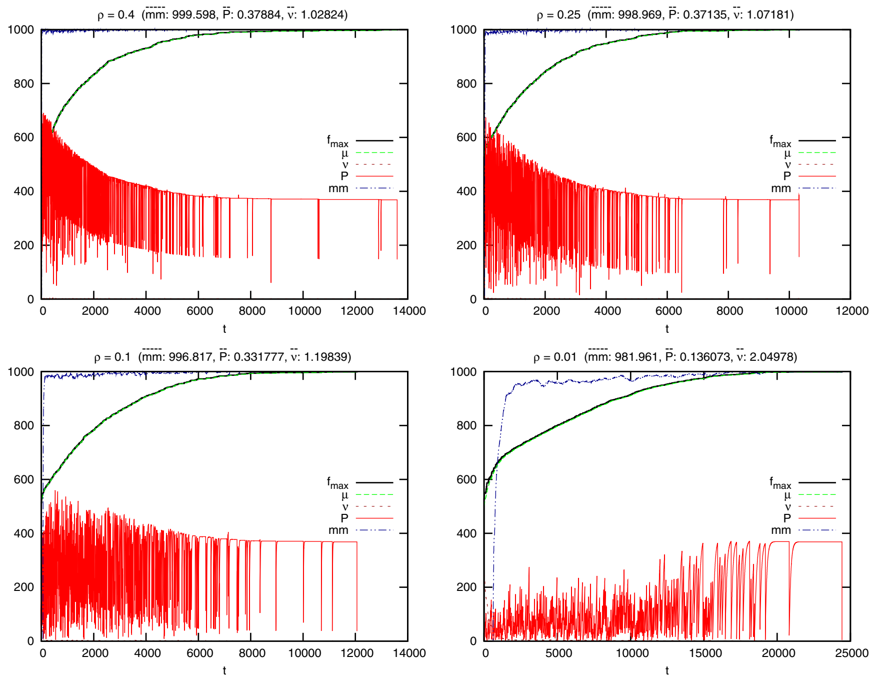


Fig. 4. Four typical runs of the MMAS on OneMax for $\rho = 0.4$ (top left), 0.25 (top right), 0.1 (bottom left), and 0.01 (bottom right)

Table 3. Indicators for the behavior of MMAS optimizing OneMax and random linear functions ($n = 1000$), as well as LeadingOnes ($n = 200$) for different ρ values

	OneMax				linear functions					LeadingOnes			
ρ	T	$\overline{mm}$	$\overline{P}$	$\overline{\nu}$	T	$\overline{mm}$	$\overline{\nu}$		ρ	T	$\overline{mm}$	$\overline{P}$	$\overline{\nu}$
1.0	16151	1000	0.38	0.999	18399	1000	1.336		1.0	34749	200	0.54	0.03
0.5	13325	999.7	0.38	1.021	17175	999.9	1.324		0.5	33481	198	0.56	0.05
0.1	11972	997.1	0.33	1.184	17454	998.9	1.356		0.1	32759	188	0.54	0.11
0.05	14054	994.2	0.28	1.344	19447	997.1	1.428		0.05	32714	181	0.52	0.23
0.01	22504	980.0	0.12	2.159	30966	987.9	1.813		0.01	32003	159	0.41	1.28
0.005	30398	971.2	0.07	2.667	44426	983.6	1.957		0.005	35719	152	0.34	2.31
0.001	75076	952.5	0.02	3.757	107027	970.2	2.509		0.001	66000	138	0.15	12.07

values. For $\rho \leq 0.045$, no run is successful, and final $P^{(t)}$ values in the 10^{-5} to 10^{-6} range show that some 100.000 iterations are necessary to find an acceptable solution (which not necessarily leads to an increased fitness).

5 Experimental Results for the MMAS

We now analyze the optimization behavior observed for the MMAS. In [8] and [9], $O(\rho^{-1} n \log(n))$ and $O(n^2 + \rho^{-1} n \log n)$ expected optimization times were proven for a variant MMAS on OneMax and LeadingOnes. This indicates that for the MMAS the optimization time does not display such a delicate dependence on ρ as previously seen for 1–Ant. Our experimental results verify this observation, but again show that the MMAS strongly imitates the optimization behavior of the (1+1) EA, in particular for the more efficient runs with larger ρ.

Our experimental results on the optimization behavior of the MMAS for linear functions is summarized in Figure 4 and Table 3. For random linear functions and OneMax, all runs, even those for relatively small ρ values like 0.005, show an optimization behavior strongly resembling that of the (1+1) EA. The average number $\overline{mm}$ of pheromone values having one of the two extremal values is above 970. In other words, in average at least 97% of the bits of the newly generated solution are determined in the same way as by the (1+1) EA. The remaining pheromone values are also close to their extreme values, as witnessed by an average variance $\overline{\nu}$ of less than 3. Not surprisingly, the optimization times are similar to the (1+1) EA-case with a considerable slow-down for small ρ values.

In Figure 4, we depict four typical runs for OneMax. We immediately notice that the four graphs are much more similar than those for 1–Ant in Figure 3, even though a wider range of ρ–values is covered.

For $\rho = 0.4$, 0.25 and 0.1 we observe an extremely short initial phase during which the pheromone values rush towards their extreme values. After 14, 37, and 114 steps, respectively, 95% of the pheromone values are $1/n$ or $1 - 1/n$. After this initial phase, in which $f^{(t)}_{\max}$ does not exceed 527 (559, 562, respectively), all indicators are similar to what we see for the (1+1) EA. This is obvious for $mm^{(t)}$, which is close to $n = 1000$ all the time, and ν, which is too small to be distinguished from the t–axis. $P^{(t)}$ differs from the (1+1) EA setting for several short periods of time. Whenever a newly generated solution different from the previous best is accepted, it takes a while for the pheromone values to move towards the extreme values. This results in the short downward dents visible in the plots. During these times, $P^{(t)}$ is smaller than in the (1+1) EA setting, but these dents end quickly and the MMAS returns to the (1+1) EA-like behavior.

The chart for $\rho = 0.01$ differs from the other three in the respect that only after $t = 2141$ iterations 95% of the pheromone values reach the extreme values. Also, $P^{(t)}$ stays below $1/e$ most of the time. Still, the value of $\overline{mm} \approx 982$ combined with a variance of $\overline{\nu} \approx 2$ indicate that during the central part of the run most pheromone values are either at their extreme values or at least very close to them.

Further experiments for smaller values of ρ show that the effects observed for $\rho = 0.01$ amplify. The variance $\overline{\nu}$ stays almost constant ($\overline{\nu} \approx 3.8$ for $\rho = 0.001$) indicating that most values of $p^{(t)}$ are close to the extreme values. The behavior of the MMAS remains highly (1+1) EA–like, only the performance drops due to the additional time needed for the pheromone values to reach the extreme values again after an update.

For reasons of space, we are not able to discuss typical runs for the test function LeadingOnes. However, the data displayed in Table 3 suffices to see that the optimization behavior of the MMAS for LeadingOnes is very similar to that of the (1+1) EA, both in terms of run-times and, more declaratively, in that we have many pheromone values at or close to the extreme values of $1/n$ and $1 - 1/n$, as witnessed by $\overline{mm}$ and $\overline{\nu}$. This fact was already observed in [9] and used to prove a lower bound on the optimization time of the MMAS. Finally, also for non-leading bits there is a strong drift of the pheromone values towards $1/n$ an $1 - 1/n$. This observation strengthens the resemblance between the MMAS and the (1+1) EA even more.

6 Conclusion

We analyzed the two existing single ant ACO approaches for three types of fitness functions. Previous research shows that, at least for certain choices of the evaporation factor ρ, both can optimize the functions OneMax and LeadingOnes with optimization times of similar order of magnitude as the (1+1) EA.

By not only regarding the resulting optimization times, but by also monitoring well-chosen theory-guided indicators during the runs of the ACO systems, we showed that whenever the optimization time was reasonable, indeed the whole optimization behavior strongly resembles that of the (1+1) EA. Our experimental investigation also complements existing rigorous mathematical analyses in that it produces actual numbers and not only orders of magnitude. Our experiments indicate that, if existent, the advantage of single ant ACO systems over classical and technically much simpler approaches has to be shown on more advanced or non-pseudo-boolean optimization problems.

References

1. http://www.mpi-inf.mpg.de/publications/index.html
2. Doerr, B., Johannsen, D.: Refined runtime analysis of a basic ant colony optimization algorithm. In: Proc. of the CEC 2007, pp. 501–507. IEEE Press, Los Alamitos (2007)
3. Doerr, B., Neumann, F., Sudholt, D., Witt, C.: On the runtime analysis of the 1-ANT ACO algorithm. In: Proc. of GECCO 2007, pp. 33–40. ACM, New York (2007)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: An autocatalytic optimizing process. Technical Report 91-016 Revised, Politecnico di Milano (1991)
5. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
6. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. TCS 276, 51–81 (2002)
7. Gutjahr, W.J.: First steps to the runtime complexity analysis of ant colony optimization. Comput. Oper. Res. 35, 2711–2727 (2008)
8. Gutjahr, W.J., Sebastiani, G.: Runtime analysis of ant colony optimization. Technical report, Mathematics department, Sapienza Univ. of Rome (2007)

9. Neumann, F., Sudholt, D., Witt, C.: Comparing variants of MMAS ACO algorithms on pseudo-boolean functions. In: Stützle, T., Birattari, M., H. Hoos, H. (eds.) SLS 2007. LNCS, vol. 4638, pp. 61–75. Springer, Heidelberg (2007)
10. Neumann, F., Sudholt, D., Witt, C.: Rigorous analyses for the combination of ant colony optimization and local search. In: van der Poorten, A.J., Stein, A. (eds.) ANTS-VIII 2008. LNCS, vol. 5011. Springer, Heidelberg (to appear, 2008)
11. Neumann, F., Witt, C.: Runtime analysis of a simple ant colony optimization algorithm. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 618–627. Springer, Heidelberg (2006)
12. Stützle, T., Hoos, H.: MAX–MIN ant system. Journal of Future Generation Computer Systems, 889–914 (2000)

Actuation Constraints and Artificial Physics Control*

Chris Ellis and R. Paul Wiegand

University of Central Florida
{chris@cs,wiegand@ist}.ucf.edu

Abstract. Swarm systems for multiagent control rely on natural models of behavior. Such models both predict simulated natural behavior and provide control instructions to the underlying agents. These two roles can differ when, for example, controlling nonholonomic robots incapable of executing some control suggestions from the system. We consider a simple physicomimetics system and examine the effects of actuation constraint on that system in terms of its ability to stabilize in regular formations, as well as the impact of such constraints on learning control parameters. We find that in the cases we considered, physicomimetics is surprisingly robust to certain types of actuation constraint.

1 Introduction

Swarm intelligence [1] is a popular and successful group of methods for controlling coordinated multiagent teams. Of such approaches, those based on variations of artificial physics models, physicomimetics [2], have particular appeal. The resulting behaviors are quite intuitive; it is easily generalized to allow for modular, heterogeneous and scalable team behaviors [3]; and traditional analytical tools from physics can be used to help diagnose and predict team behaviors[2]. Physicomimetics is particularly well-suited for tasks that require stable geometric formations such as lattices or rings, and under the proper circumstances one can show that teams will settle into "low-energy" positions provided by such structures.

It is clear that control methods based on artificial physics models are performing two essentially different tasks: 1) *predicting motion* of particles within a particle-based physics model (particle model), and 2) *producing control input* to move agents in some real or simulated world (environment). When agents are treated as simple point-mass particles with no additional constraint on their motion, these roles do not conflict. However, for realistic control systems operating in the environment (e.g., nonholonomic robotic platforms), a conflict between these roles occurs when the agents being manipulated cannot move as requested. Analyses regarding the stabilization of regular formations, for instance, rely on a temporal element — the dynamics of the particle model itself. When there is a disconnect between model prediction and control, such analyses are questionable since the dynamic will almost certainly differ, potentially quite radically. Additionally, though parameters for physicomimetics control systems are often hand-coded, complex problems require some kind of learning, and it isn't clear how the disconnect between prediction and control affects the learning gradient.

* This work was funded as part of the ARL/HRED project on *Team Performance in Human-Agent Collaboration* (W911NF-07-R-0001).

Still, physicomimetics has been successfully applied to a wide range of control problems including mobile robot formation [2], multi-robot chemical plume tracing [4], and heterogeneous, multiagent in-port ship protection [3]. Moreover, in many cases control systems have been demonstrated both in simulation and on physical devices, where actuation constraint varies widely.

We show artificial physics based control systems *are* affected by actuation constraints on the agents; however, physicomimetics is surprisingly robust to such prediction / control disparities. We construct a common nonholonomic control system that constrains agent motion in a number of ways and parameterizes the maximum allowable turning speed and examine the effect of this parameter on lattice formation. Considering a more complex covert tracking problem that requires learning, we discover that added constraints affect properties of the system and influence the learning gradient. Only in extreme cases are the behaviors qualitatively different.

The next section will discuss the control system we are using in detail. Section three will discuss the effects of constraints on simple hexagonal lattice formation, while section four will detail our efforts to learn physicomimetics based control solutions for a covert tracking problem. We finish up with a short discussion of related work, as well as our conclusions and future plans with this research.

2 Representing Agent Behaviors with Artificial Physics

2.1 Physicomimetics

Physicomimetics provides a framework for the control of multiple agents [2]. Each agent has its own physicomimetics model, which is updated at each time step from that agent's knowledge of its environment, including the observed positions and velocities of all observed agents. Agents are treated as point-mass (m) particles. Each particle has a position, $\mathbf{x}$, and velocity, $\mathbf{v}$. We use a discrete time simulation, with time step Δt. At each time step, the particle is repositioned based on the velocity and the size of the step, $\Delta \mathbf{x} = \mathbf{v}\Delta t$. The change in velocity of the particles is determined by the artificial forces operating on the particles, $\Delta \mathbf{v} = \mathbf{F}\Delta t/m$, where $\mathbf{F}$ is the aggregate force on the particle as a result of interactions with other particles and the environment. Each particle also has a coefficient of friction, $c_f \in [0, 1]$. Velocity in the next step becomes $(\mathbf{v} + \Delta \mathbf{v})c_f$, stabilizing the system [2]. When this new velocity is computed in the physicomimetics model, the agent tries to the best of its ability to match it in its own environment. The values of these masses, frictions, and force laws are what govern the motion in this system, and are either hand selected or learned in some fashion.

There are two constraints: the magnitude of the force cannot exceed F_{max} and the magnitude of the velocity cannot exceed V_{max}. These restrict acceleration and velocity of particles in the model. Also, since there is an emphasis on *local* interactions, there are further restrictions on the range of effect particles have on each other.

Advantageously, a variety of force laws can be employed to different effect. The parameters of the above model, coupled with the force law parameters, provide engineers with mechanisms to adjust the behaviors of agents. Finally, since physicomimetics is based on physics, practical analyses are possible using traditional physics techniques such as force balance equations, conservation of energy and potential energy [2].

A slight variation of the well-known Newtonian force law will be used in this paper:

$$F_{ij} = \begin{cases} -G\frac{(m_i m_j)^a}{r_{ij}^d} & \text{if } r_{ij} \in [0, R) \\ G\frac{(m_i m_j)^a}{r_{ij}^d} & \text{if } r_{ij} \in [R, E] \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The magnitude of the force is determined by choice of gravitational constant, G. The force law repels particles closer than R and attracts particles past that distance but within the range of effect, E. The gradient of the force can be controlled using d, and a can raise or lower the importance of mass on the force. In total, there are two parameters associated with each particle (m and c_f) and five parameters associated with their interactions (G, E, R, a, and d). Distance variable r_{ij} is an observed phenomenon.

2.2 Constraining Agent Motion

Our goal was to simulate a parameterized differentially steered device. We used a number of parameterized constraints on agent movement. Physical constraints such as maximum velocity V_{max}, maximum acceleration a_{max}, and maximum turning speed θ_{max} are placed upon the agents, and the physicomimetics control system is allowed to suggest velocities without regard for these limits. At each time step, the agent will update its orientation, velocity, and position according to the following algorithm:

1. The orientation and velocity of the agent is rotated by $\Delta\theta$ towards the suggested velocity, where $\Delta\theta = min\left(\theta_{max}, \theta_\delta\right)$. θ_δ is the difference in orientation between the current agent orientation and the orientation of the suggested velocity.
2. The magnitude of the agent's current velocity is set to $|\mathbf{v}| = |\mathbf{v}_{prev}| \cdot cos(\Delta\theta)$, where $|\mathbf{v}_{prev}|$ is the magnitude of the velocity at the previous time step.
3. The suggested velocity is projected along the updated orientation vector, and agent velocity is updated to as close to the projected suggested velocity as a_{max} permits.
4. The magnitude of the agent's velocity is constrained by the maximum velocity, $|\mathbf{v}| = min\left(|\mathbf{v}|, |V_{max}|\right)$.
5. Agent position is updated according to the new computed velocity, $\Delta\mathbf{x} = \mathbf{v}\Delta t$.

Dampening the speed of the agent proportionately to $\Delta\theta$ in step 2 has a stabilizing effect on the agents, which we use here in place of friction. It is for this reason that an agent with a turning speed constraint of π is different from a traditional agent. Of the thresholds discussed, the turning speed constraint θ_{max} was chosen as the independant variable in order to observe to what degree increasing constraints impact the performance of both the agent and learning algorithms operating on that agent.

3 Constraining Motion in Simple Lattice Formations

One of the simplest and most natural formations obtainable by agents controlled via physicomimetics is an hexagonal lattice. Straightforward swarm design methods can produce solutions capable of settling into a regular isometric grid quickly and efficiently. With appropriate parameters, one does not even need friction for the system to find equilibrium in such stable configurations because the low potential energy wells of the system correspond with these structures.

We refer to our control group as the “traditional model”, described in section 2.1, where there is no inconsistency between the environment and the particle model. In this case, a swarm designer can affect lattice width via the R (attraction-repulsion boundary) parameter. The effect range, E, is typically set at $1.5R$ to be less than the $\sqrt{3}R$ factor that allows second-tier points in the lattice to be visible. Our parameters follow those of [2]: $R = 50, E = 75, G = 1200, a = 1, d = 2$, save that we use 100 particles (Spears used 200), and we do not use friction. This system will settle into a hexagonal lattice.

We investigate two properties of the system: *settling time* and *lattice quality*. Settling time is the time it takes the system to find a quiescent state. Lattice quality is a measure of how faithfully the distributed agents replicate an isometric formation.

3.1 Settling Time

Under the right conditions, a particle model will lose energy as it converges on a stable formation. When agents cannot move as dictated by their surrogate particles, unstable dynamics might be introduced in the physicomimetics system since the criteria of the proofs for stability in [2] are not all met. To test this, we considered control systems with varying constraints on turning speed ($\theta_{max} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 3.14\}$ radians), as well as the traditional (non-constrained) model. We ran each model 30 times and analyzed the dynamics in terms of the average scalar acceleration magnitude of all agents each step.

Using our nonholonomic control system above, we find that the system not only settles into a formation in all cases, but the damping factor for sharper turning seems to help the system settle faster. The left panel in Figure 1 below illustrates averages over the thirty trials of the settling behavior in four of the above groups. We examined all of the above groups, and the basic curve characteristic is similar in all cases, and standard deviations (not shown) indicate very little variability in this settling behavior.

To investigate this behavior more carefully, we consider the *settling time* of the system: the number of time steps taken for the average magnitude of acceleration to drop below an empirically selected threshold, 0.01 distance units per step squared. The right side of Figure 1 shows these results for all experimental groups. Pair-wise t-tests using Bonferoni adjustment indicates no statistical differences between any of the constrained groups, but all constrained groups have a significantly lower settling time than the traditional model (95% confidence).

While constraining the motion of the agents undoubtedly affects the rate at which they settle into a stable formation, our nonholonomic constraints do not prevent this ability in general. Indeed, our differentially steered agents settle *faster* than the traditional approach because of the damping influence on sharp angle motions.

3.2 Lattice Quality

Arriving at a stable configuration quickly does not necessarily imply that the same configuration is reached. Again, swarm design on the traditional system to produce hexagonal lattices is predicated on a basic understanding of traditional physics. This understanding is of questionable value when particles cannot move freely.

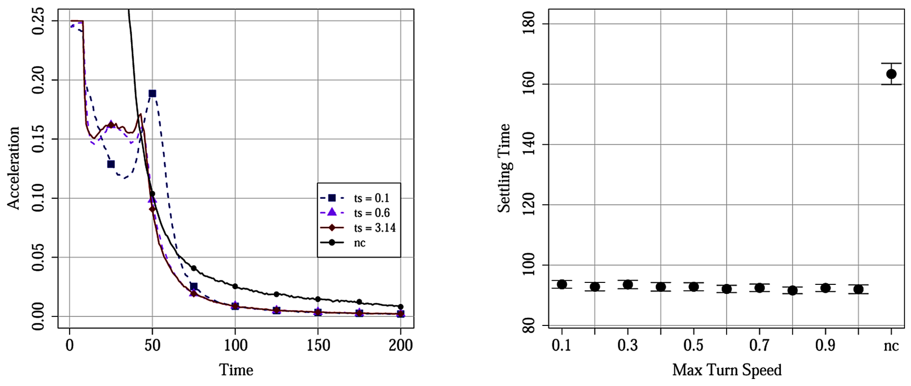


Fig. 1. *Left graph*: four systems settling into a formation. Each curve is the average scalar acceleration magnitude of the system across thirty different simulations. *Right graph*: the settling time for each of the groups. Points and wings represent means and 95% confidence intervals.

To investigate this, we again run the above experimental groups ($\theta_{max} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 3.14\}$ and traditional case) for thirty trials apiece and measure the average lattice quality for fifty steps after the system has settled (acceleration magnitude has dropped below 0.01). Peaks in the acceleration graph represent the moment when agents are slipping into their minimal energy configurations. Peak position is likely a function of the number of agents.

We found the lattice quality measure used in [2] too sensitive to the value of R for our purposes. Instead, we compute the Delaunay triangularization [5] of the particles, then measure the coefficient of variation in edge lengths in this graph. To reduce boundary effects, we use only edges with points in the inner 85% of total area covered by the agents. Figure 2 below illustrates the mean and confidence intervals for the quality results for all groups.

Using the multi-way comparison previously described, we find that the lattice quality of the traditional group is significantly better than the constrained cases. Also, the $\theta_{max} = 0.1$ case differs from all other cases. The 0.2 case differs from the 0.6, 0.7, 0.8, 0.9, and 3.14 cases. All other comparisons are statistically indistinguishable.

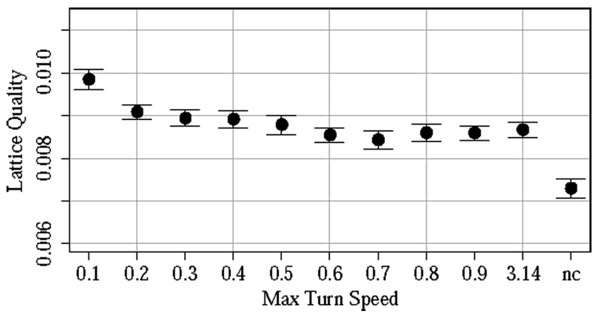


Fig. 2. Lattice quality for each group. Points and wings: means and 95% confidence intervals.

While our nonholonomic agents form hexagonal lattices that are of marginally lower quality than the traditional models, there is no doubt that the lattices *are* formed. Visually, they look very similar. A random dispersion of 100 points results in a lattice quality of more than four times that of the worst of our groups; all groups formed lattices that were significantly better from a statistical point of view. Moreover, the lattice quality is fairly robust to the *degree* of constraint (in terms of θ_{max}): using constraints marginally decreases lattice quality, but the degree of constraint is not particularly important.

4 Constraining Motion in the Covert Tracking Problem

We are interested in a generalized form of a multi-target tracking problem, where our coordinated team of agents learns to distribute the task of tracking various targets with differing capabilities and with different objectives in mind. Such problem domains require fairly sophisticated swarm designs and (typically) some kind of learning.

However, it isn't clear how portable swarm design methods are when there are profound disconnects between how agents *can* move and how a swarm system *directs* them to move. Moreover, it isn't clear how such constraints impact learning performance. To begin to answer this question, we focus on a simple form of our more general problem and investigate how our constrained nonholonomic controller impacts learning, as well as the final solution quality.

4.1 Covert Tracking

For this experiment we use a single tracker and a single target. The goal is for the tracker to follow the target as closely as possible and not be seen by the target. The target has a field of vision that consists of two concentric circles. The target will detect the tracker if it is anywhere in the inner circle, 10 distance units. The outer radius (30 distance units) is a $\frac{3}{2}\pi$ radians arc, such that the target has a "blind spot" directly behind its facing direction. The target moves at a maximum velocity of 1 unit per step and is constrained in a way similar to the tracker with a $\theta_{max} = 0.05$. It randomly wanders as follows: with a probability of 0.05 each time step, it selects a new desired facing direction and changes its velocity to match that preference (allowing for actuation constraints). The new direction is chosen uniformly at random within a relative ± 3 radians. The behavior is fairly smooth, with occasional surprising turns.

Initial positions of the target and tracker are chosen uniformly at random in a rectangular area of 80×60 units in size. The tracker has a 2π radians field of view up to 80 distance units and has a maximum velocity of 2 units per step. It's behavior is controlled via physicomimetics as described above using three types of particles. The first represents the target, the second the tracker, and the third a *virtual particle* — a particle representing an unembodied concept to be used by the control system, described below. Each step, the tracker constructs a particle model, placing a tracker particle in its own position, a target particle in the position of the target (if it is seen), and a virtual particle in a position computed using a relative range and bearing from the position of the target. The offset bearing is relative to the bearing of the target, and the tracker estimates the target bearing based on its change in position.

The virtual particle is necessary if we hope to have the trackers exploit the blind spot of the target. While our representation does not prescribe how this particle is used, the most obvious solution is to place the particle in the blind spot. The learning system is responsible for determining this.

The control system requires 23 parameters. The mass and coefficient of each of the three particles (6), the force law parameters for each interaction ($5 \cdot 3$), and a range / bearing offset for computing the position of the virtual particle. These are all represented as real values. The table below describes the permitted ranges for these values.

Table 1. Ranges for control system parameters, $\epsilon = 0.00001$

Type of parameter	Range	Type of parameter	Range
mass, m	$[0, 200]$	distance power, d	$[-10, 10]$
friction, c_f	$[0, 1000]$	mass power, a	$[-10, 10]$
effect range, E	$[0 + \epsilon, 480]$	virtual particle range, ρ_v	$[0, 60]$
AR boundary, R	$[0 + \epsilon, E]$	virtual particle bearing, θ_v	$[-\pi, \pi]$
gravity, G	$[-1000, 1000]$		

4.2 Learning a Physics Model for Control

The 23 parameters just discussed were encoded into a real-valued genome, and a (5+35)-ES was used for optimization. Gene values were in the range $[0.0, 1.0]$, and were scaled to the ranges of each individual parameter of the force laws the physicomimetics control system. Fitness for an individual was aggregated over 20 trials, for 600 timesteps per trial. Adaptive mutation was employed as in [6] with $\sigma_{init} = 0.25$ and $\sigma \in [0.005, 0.25]$. Evolution took place over 50 generations, and the most fit individual found was then evaluated with a number of metrics for our empirical analysis (described below).

Fitness at each time step for each individual was evaluated as follows.

$$F(a \in trackers, b \in targets) = R_\alpha \cdot \left(sees(a,b) \frac{D(a) - r_{a,b}}{D(a)} \right)^{R_\beta} - P_\alpha \cdot sees(b,a)$$

$$sees(a, b \in agents) = \begin{cases} 1 \text{ if agent } a \text{ 'sees' agent } b \\ 0 \text{ otherwise} \end{cases}$$

Here $D(a)$ represents the vision range of agent a. There are three parameters R_α, R_β, and P_α, used here to tune the ES towards different desired behaviors. R_α is the reward scaling factor, R_β is an exponent that changes the signifigance of the distance of the trackers to the targets they see, and P_α is the penalty scaling factor. For the purposes of our experiment, $R_\alpha = 1$, $R_\beta = 1$, and $P_\alpha = 3$. This creates an environment where the fitness reward increases linearly with the proximity of the tracker to its target. The reward given per time step can be up to 1.0. A flat penalty of 3 is applied at each time step if the tracker is seen by the target. Increasing the reward for smaller distances causes the tracker to get as close as possible, while the penalty for being seen ensures that the tracker will avoid the vision area of the target.

4.3 Predicting vs. Controlling Agent Behavior

The first hypothesis that needs to be confirmed is that limitations on agent mobility create a disconnect between the executed behaviors of the controlled agents and the described behaviors of their analogous particles. If that is so then presumably the learning system will have to work with, or compensate for, those differences. Our suggested position difference measure, Δ_{pos} confirms the first part of this reasoning.

At each time step the physicomimetics system suggests a velocity that the agent, to the best of its ability, tries to execute. This yields a suggested position, $\hat{\boldsymbol{x}}$, and the actual updated position the agent is capable of achieving, $\boldsymbol{x}$. The suggested position difference for a given time step is then the Euclidean distance between the suggested and actual position for that step. The magnitude of this value represents the degree to which the agent is incapable of matching the directions given by the control system. Our measure aggregates this value over all time steps according to the following equation: $\Delta_{pos} = \frac{\sum_{t=1}^{T}(\|\hat{\boldsymbol{x}}(\boldsymbol{t})\|-\|\boldsymbol{x}(\boldsymbol{t})\|)}{T}$.

The Δ_{pos} for the best of run for each of thirty trials of the EA were collected. The results were aggregated over each value of θ_{max}, and a Bonferroni-adjusted t-test was applied to test for statistical differences between the results (see the left graph of Figure 3). As is expected, as the turn speed constraint is relaxed, the overall trend of the output of the physicomimetics model moves to more closely match the actual change in location of the agent. However, even as the turn speed approaches the point where an agent may turn any direction in a single time step, there is still a statistically significant difference compared to the traditional agent. As expected, the constraints create a disconnect between particle model prediction and the resulting executed behavior.

4.4 Effects on Learning Performance

Despite the fact that there is the disconnect discussed above, the learned solutions are surprisingly good. Though the learned solution in the traditional case is significantly better than all the others (in fact, all groups are significantly different from one another), all but the most extreme cases learn the same basic behavior: Set the virtual particle inside the blind spot of the target, then track the target from that position.

Consider the right plot in Figure 3, below. First, all values are reported in terms of their average fitness per step. Second, the means and confidence intervals of the best of run values for each experiment group are plotted. Finally, an *incursion zone* is plotted as a shaded rectangle to highlight the efficacy of the learned solutions. This zone represent distances smaller than the range of vision of the target, scaled to the limit of the tracker's range of vision $\left(\frac{D(a)-r_{a,b}}{D(a)}\right)$. Fitness values within that range must result from behaviors that stay within the outer range of the target's range of vision and receive no penalty from being seen (i.e., in the target's blind spot). The wider confidence wings on the 0.15 and 0.3 groups occur because some trials fail to discover this, while others succeed. The learning gradient for placing the virtual particle becomes particularly steep when the constraint is high.

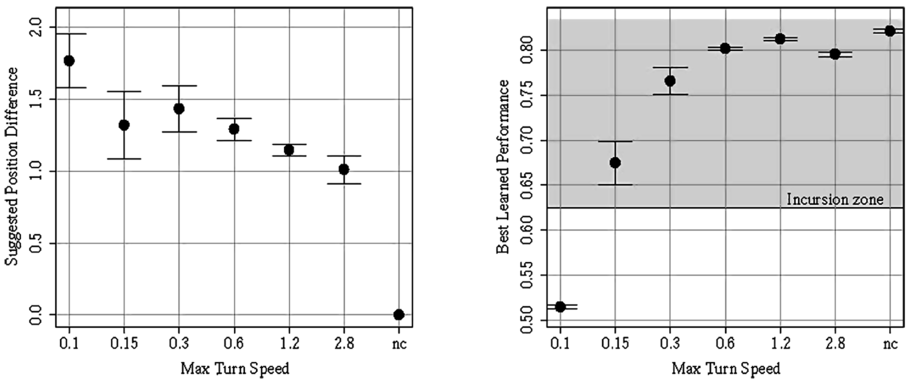


Fig. 3. *Left graph*: Suggested position difference for several experimental groups. *Right graph*: Best of run fitnesses for all groups. The grey area represents values where trackers remain (on average) in the blind spot of the target. Points and wings: means and 95% confidence intervals.

5 Related Work

There is an increasing wealth of literature that uses swarm intelligence for coordinated control of groups of agents, such as physicomimetics [2,3,4], methods based on social potential fields [7], or methods based on flocking and schooling behaviors in animals [8]. These approaches rarely focus on the effects on actuation constraints on agents under control, though the agents themselves are often nonholonomic.

Additionally, there are traditional control theory methods for formation control in agents, typically based on a leader / follower paradigm [9]. In an interesting middle-ground approach, [10] presents a method for designing cooperative formation control systems for groups of mobile robots (both holonomic and nonholonomic) based on potential functions. [11] examines three aspects of a behavior-based approach to coordinated multiagent control that includes control of robots under differential steering. These methods incorporate explicit notions of actuation constraint and (often) include formal justifications for which certain patterns will stabilize; however, they lack the intuition and simplicity of bottom-up, nature-based approaches.

6 Conclusions and Future Work

Though it is clear that actuation constraints can have potentially profound impacts on how a multiagent system must be controlled to produce useful coordinated behavior, little effort has been made to date to determine how swarm-based approaches are affected by such constraints. We believe a closer look at when and how actuation constraints affect swarm-based behaviors is therefore justified, and this paper presents a preliminary empirical look into these effects in two simple domains: hexagonal lattice formation and covert tracking. We examined a differential-like control system that allowed us to vary the degree of constraint on agent movement.

We found that while constraints do impact system performance, only in the most extreme cases were the physicomimetics control methods unable to accomplish the basic tasks at hand. The systems did not destabilize, nor did they alter the basic character of the behavior implemented in the unconstrained cases. Simple, high-quality hexagonal lattices were formed even when maximum turn speed was quite low, using the same parameters as successful unconstrained physicomimetics agents used to solve the same problem. Constraints affected learning performance only minimally, save when they were particularly severe. Our results provide some hope that for certain, simple problems physicomimetics is fairly robust to these kinds of mobility limitations.

We believe that even when the constraints are severe, learning difficulties can be mitigated using transfer learning. In the most extreme case of the covert tracking problem, the system never learns to place the virtual particle in a useful position and the agent learns simply to stay outside the vision range. Our approach is to first learn the control parameters in the traditional way, then use this to bias the search for behaviors when the tracker cannot move so freely. Early results are encouraging.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence. Oxford University Press, Oxford (1999)
2. Spears, W., Spears, D., Hamann, J., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous Robots 17, 137–162 (2004)
3. Wiegand, R., Potter, M., Sofge, D., Spears, W.: A generalized graph-based method for engineering swarm solutions to multiagent problems. In: Parallel Problem Solving From Nature, pp. 741–750 (2006)
4. Zarzhitsky, D., Spears, D., Thayer, D., Spears, W.: A fluid dynamics approach to multi-robot chemical plume tracing. In: Auton. Agents and Multi-Agent Systems, pp. 1476–1477 (2004)
5. Guibas, L., Stolfi, J.: Primitives for the manipulation of general subdivisions and the computation of voronoi. ACM Trans. Graph. 4(2), 74–123 (1985)
6. Schwefel, H.P.: Numerical Optimization of Computer Models. Wiley, Chichester (1981)
7. Reif, J.H., Wang, H.: Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots. Robotics and Autonomous Systems 27(3), 171–194 (1999)
8. Reynolds, C.: Flocks, herds and schools. Computer Graphics 21, 25–34 (1987)
9. Ogren, P., Egerstedt, M., Hu, X.: A control lyapunov function approach to multiagent coordination. IEEE Transactions on Robotics and Automation 18, 847–851 (2002)
10. Nguyen, D., Do, K.: Formation control of mobile robots. International Journal of Computers, Communications and Control I(3), 41–59 (2006)
11. Lawton, L., Beard, R., Young, B.: A decentralized approach to formation maneuvers. IEEE Transactions on Robotics and Automation 19(6), 933–941 (2003)

Genetic Repair for Optimization under Constraints Inspired by *Arabidopsis Thaliana*

Amy FitzGerald and Diarmuid P. O'Donoghue

Department of Computer Science, NUI Maynooth, Co. Kildare, Ireland
amyfg@cs.nuim.ie, diarmuid.odonoghue@nuim.ie

Abstract. It has recently been proposed that the model plant, *Arabidopsis thaliana* (thale cress), uses a newly discovered genetic repair system to repair errors at the genetic level. *A. thaliana* uses information from the grandparent's genes as a basis for this correction – so genetic information appears to skip a generation. We apply this gene repair strategy to a combinatory optimization problem, firstly comparing the performance of parent and grandparent based repair. Subsequent experiments expand our understanding of the GeneRepair algorithm, by examining the parameters of fitness and direction involved in the generepair process. Our results point to a tentative explanation as to why *A. thaliana* might have evolved such an apparently complex inheritance process.

Keywords: Evolutionary optimization, genetic repair, constraints, Arabidopsis thaliana.

1 Introduction

Evolutionary Optimization (EO) is an optimization strategy that is inspired by Darwin's idea of survival of the fittest. EO effectively implements a "generate and test" beam search to find near optimal solutions to complex problems, such as NP-Complete problems, in a reasonable amount of computing time. A population of candidate solutions are created and allowed to converge towards a global optimal under the guidance of a suitable fitness function. Evolutionary strategies are effective in exploring complex solution spaces, where each individual explores part of the search space. However, EO is less suited to enforcing validity constraints [1] on these search spaces.

Evolutionary optimization and related approaches, use biology as their inspiration. This paper turns again to the biology domain, looking at some recent advances in the study of the *Arabidopsis thaliana* (thale cress) plant. *A. thaliana* appears uses a genetic repair process to repair errors in its genes. This repair process uses genetic information originating in the genes of the grandparent – information which does not appear to be detectable in the genes of the parent.

This paper presents a comparison of these repair strategies, on a standard combinatoric optimization problem. Results for a biologically inspired penalty points technique [2] act as a benchmark. The results of our initial experiment and presented and discussed, followed by two supplementary experiments that clarify some issues raised by the first experiment.

To the best of our knowledge, no authors have previously examined the effectiveness of grand-parent based repair in evolutionary computation, or compared parent based and grandparent based approaches to genetic repair.

2 Gene Repair in Arabidopsis Thaliana

Arabidopsis thaliana (thale cress) is a model plant used for a wide variety of detailed studies and was the first plant genome to be sequenced. The Arabidopsis plant has one of the smallest genomes with about 157 million base pairs and five chromosomes. The Arabidopsis genome encodes 27,000 genes and 35,000 proteins.

Lolle *et al* [3] investigated *A. thaliana* plants with an organ fusion mutation on the Hothead gene (HTH), resulting in an abnormal formation of the plant's flower. Their studies revealed that two plants with the HTH can produce offspring without this abnormality, forming perfectly normal plants. The resultant offspring have the normal form of the hothead gene (hth), even though this information was present in neither of the parent's genomes. That is, approximately 10% of the offspring were found to revert to the normal form of the hothead gene, which is a far higher rate than can be explained by random mutation of these specific alleles (which would be of the order of 1 per billions per allele per generation). It was found that that these revertant genomes all appeared to inherit genetic information from their grand-parents genomes, which had the normal (hth) form. Thus, genetic information appeared to skip a generation, reappearing in a subsequent generation. In an interview with the Washington Post (March 23rd, 2005) Robert Pruitt referred to this as a "parallel path of inheritance", which appears to occur in addition to standard Mendelian inheritance. In essence, a corrective template is used to correct broken or damaged sequences of DNA, possibly in response to stress placed on the plant due to the presence of a genetic mutation.

While Lolle's controversial [4] explanation relies on a cache of RNA inherited from previous generations, we focus on the explanation offered by Ray [5] that is compatible with Lolle's findings. Ray's explanation relies on an archival form of DNA, that serves to store the ancestral DNA but which is not detected by the processes used to sequence the regular encoding of DNA.

Thus, in our implementation each individual maintains its own archive of 2 generations of ancestral genetic information. This yields a custom made repair template for each individual in the population (see Figure 2).

3 The TSP Evolutionary Optimization Problem

To examine the performance of various GeneRepair strategies, we used the standard problem called the Traveling Salesman's Problem (TSP) (or the Hamiltonian Circuit problem). This NP complete problem involves finding the shortest path that visits each of a number of vertices (cities), visiting each just once and returning back to the original vertex (city). The TSP problem is thus a minimization problem, where the best results correspond to a lower tour length. In this paper we use the results generated for the 51 city traveling salesman problem (eil51) from the standard TSPLib problem set. We point out that our focus was on comparing the effectiveness of GeneRepair strategies and not on producing short tours for this problem set *per se*.

One specific requirement for the problem domain was that it has identifiable validity constraints. That is, invalid solutions to this problem can be generated and can be identified. A TSP solution is invalid if the tour does not visit all cities, if a city is visited twice or if the tour does not return to the starting city.

The mutations of *A. thaliana* studied by Lolle *et al* [3] were from living plants, which were thus viable plants. There are a relatively small number of known viable (living) mutations of *A. thaliana*, corresponding to a tiny fraction of combinations of its 157 million base pairs. In contrast, the TSP does not have such viable mutants as all mutants form invalid (non-viable) solutions to the problem. These non-viable solutions are repaired immediately, whereas *A. thaliana* does not appear to involve genetic repair until the next generation. While our experiments appear to involve a slightly more pro-active gene-repair process, it was considered that is was not a very significant difference. These differences may perhaps lie more in the environmental stress factors that trigger gene repair in *A. thaliana.*

All experiments were run with the same experimental set-up, where only the described parameters were changed between experimental conditions. Initial experiments were conducted with a population size of 500 for 500,000 generations. This yields an overall search space that examines 250,000,000 different possible tours. We point of that this is a tiny fraction of the total search space of approximately $1.5.*10^{64}$ possible tours. Several independent runs were conducted for each experimental condition, to counteract against the randomized nature of EO. (A computer cluster was used to support simple independent simulations).The best results produced at each stage were recorded, as well as the generation at which those results were generated. The best and average results are presented in the next section.

4 Evolutionary Optimization with GeneRepair

This paper applies the genetic repair process described by Lolle [3] and Ray [5] to an otherwise standard EO algorithm (with unmodified crossover and mutation operators). The GeneRepair process is largely independent of the application domain itself. The only influence the problem domain has is through the genetic strings of the ancestor population. Thus, we conclude that this repair process is (largely) domain independent and may work as well or even better on a variety of other problem domains. This may be related in some way to the findings of Lolle *et al* [3] who found that gene repair in *Arabidopsis thaliana* appeared to operate throughout the DNA sequence and thus appear to be a general mechanism for extra-genomic inheritance.

However, before examining the GeneRepair process itself, we must first look at the underlying EO algorithm.

4.1 Representation

Each allele in our EO algorithm encodes a single city and each city is uniquely encoded. Therefore there is a 1-to-1 association between cities of the TSP problem and the city's representation within the EO algorithm. Solutions to the TSP are formed as an ordered list of cities and the entire population is composed of a fixed number of individual tours (see Figure 1). Tours are stored as a fixed length and ordered list of cities (the number of cities in TSP determining the length of representation). So, the relative order of cities determines their position within a tour.

This representation allows two types of genomic error to occur to individuals within a population. Firstly, duplicate errors may occur when a city is repeated within a candidate tour in the population. Secondly, omission errors occur when a city is absent from a candidate solution in the population. We highlight that error is a violation of the solution constraints as required by the TSP. Because of our fixed-length encoding, omission and duplicate errors are always found in pairs. Thus, an omission error is always has a corresponding duplicate error. As can be seen in Figure 1, duplication of the "2" causes omission of "6" from the genetic sequence. Repairing such errors shall be discussed in Section 4.2 below.

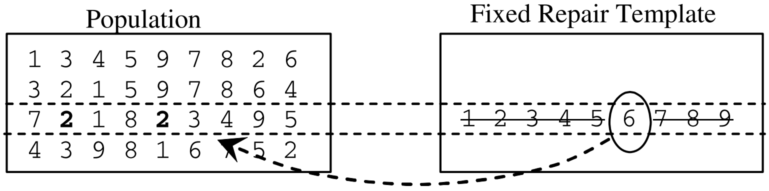


Fig. 1. The third Individual has duplicate and missing information, which must be repaired

4.2 Fitness Function, Crossover, Mutation

Before we present the GeneRepair operator, we first clarify the structure of the EO that is working in conjunction with GeneRepair. We now briefly describe the operators of fitness evaluation, selection, crossover and mutation rates. We point out that these are all generic operators, none of which are tailored to the given problem domain (see [6] for a discussion of specialized operators). The fitness function operates on individual tours, calculating the Euclidean distance between each city pair in turn, returning the sum of the individual inter-city distances.

Previous work on the GeneRepair operator has investigated the performance of parent based repair [7, 8]. This was compared to the performance of a variety of alternative strategies for implementing constraints on EO. In particular, this work explored how GeneRepair interacted with the standard evolutionary parameters, especially mutation rate and crossover mechanism.

While the results of Mitchell [7] indicate that best results are produced using Tournament selection, the results in this paper used Truncation selection with a truncation factor of 2. Thus, at the end of every generation the fittest half of the population we replicated and replaced the less-fit half of the population. The decision to use Truncation selection was made because of its simplicity and because it made detailed analysis of results (not discussed here) easier to conduct. Similarly, single point crossover was used to create new individuals. Thus, a random point on the genetic sequence of both parents is chosen, the first portion of the first parent and the last portion of the second parent are combined to form the new individual (solution).

Mitchell [7] indicates that GeneRepair requires a relatively low rate of (point) mutation - 2% of the alleles in the population are mutated on every generation. This low rate of mutation may be explained because the GeneRepair operator has a mutagenic effect, meaning that the background level of mutation can be somewhat lower than may otherwise be expected.

4.3 The GeneRepair Adjunct Operator

The GeneRepair operator is used in this paper to ensure that all solutions in the population are valid – that there are no omission or duplication errors in any of the solutions stored in the population. Such error can be generated from two different sources. Firstly, the crossover operator combines genetic information from two individual to create a new individual. We use single point crossover that chooses a single point along the allele sequence of both parents, combining the first half of one parent with the second half of the other parent. Thus a new individual is formed.

Genetic errors are identified when the genetic information of newly generate offspring violate the (mathematical) constraints of the TSP. We identify on two categories of error: *omission errors* and *duplication errors* (as discussed in Section 4.1 above).

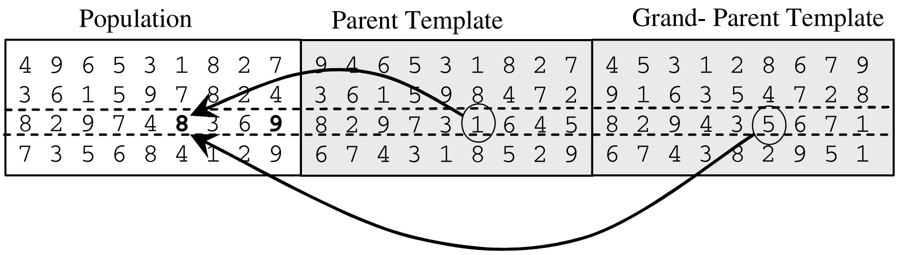


Fig. 2. Does Parent or Grandparent based Correction yields better results?

Mitchell [7] and Mitchell *et al* [8] and others [9] examined several biologically and non-biologically inspired templates, but did not explore the use of a grandparent based repair template.

4.3.1 Template Origin

Our first objective was to compare the performance of parent based GeneRepair with that of grand-parent based GeneRepair. Template driven GeneRepair operates in two phases as follows. The first phase (called error detection) identifies all occurrences of duplicate errors in the current population. In our first experiment, these duplicate errors were identified in a fixed left-to-right manner. So the second and subsequent occurrences of cities within a tour are detected as errors and are sent to the second phase, called correction. (This left-to-right decision shall be addressed further in Section 4.3.2 below.) These duplication errors can be seen as the bold figures in the Current Population of Figure 2 above.

The second phase (called error correction) of GeneRepair repairs the identified errors. While each individual was being examined, the cities of the current population are tagged in the parent and grandparent populations. Thus, un-tagged information in both populations form an ordered list of missing cities. These missing cities are used to replace the duplicate cities in a left-to-right manner.

The first experiment compared the effectiveness of parent and grandparent based GeneRepair, on the TSP problem described above. Table 1 summarizes these results, showing the shortest tour identified across these experiments and the mean results produced by each strategy.

Table 1. GrandParent based GeneRepair outperforms parent based repair

	Min	Mean
Parent Strategies	505.43	549.43
GrandParent Strategies	491.18	548.24

As shown above, the grandparent strategy far outperformed the parent strategy on these experiments. In fact, all grandparent based results outperformed all of the parent based results. Additionally, the relatively high mean of the Grandparent based repair was due to one particularly poor result of this strategy.

Not only did grandparent based repair generate better results, it did so in significantly fewer generations that the parent strategy. The grandparent strategy reached a result within 15% of the optimal in 5,500 generations while the parent strategy reached a result within 19% of the optimal in 8,350 generations. Also, we point out that our focus was on comparing strategies inspired by *Arabidopsis thaliana* and little effort went into tailoring our EO to generate good results for this problem set.

An explanation for the superior performance of grand-parent based repair, we turn to the differences between the offspring and its parent and grandparent. We point out that the grandparent has a higher probability of being *different* to the individual being repaired than its immediate parent. Thus, grandparent based repair generally has a larger disruptive effect on the individual than parent based repair. As our EO converges, the diversity in the population tends to reduce so that there is little difference between parent and offspring. (Mutation or even adaptive mutation is often used to counteract this tendency, allowing convergence to a global rather than a local optimum). Thus, we theorize that the grandparent proves to be a better template for repair than the parent, because of its potential for greater dissimilarity with the individual. This conclusion suggests that great-grandparent based repair should further outperform grandparent based repair – this being the subject of our current work. However, we do expect a decreasing pay-off as additional generations are archived in the repair process. As with Arabidopsis thaliana, it may well be that the additional expense of adding generations may not produce a commensurate payback in performance.

Another interesting observation arose from our analysis of these experiments. When a single occurrence of a duplication error is identified, both parent and grandparent strategies will generate the same *new* repaired individual. So when converging towards a global optimum for the given problem, we might expect fewer errors and thus less of a difference between parent and grandparent strategies. Therefore, much of the difference between these two strategies will occur earlier in the evolutionary process.

4.3.2 Direction of Error Detection

The next experiment attempted to assess the impact that the direction of error detection has upon solution quality. In addition to the left-to-right error identification strategy, two other strategies were investigated: right-to-left and random direction. The left-to-right and right-to-left were fixed throughout whereas the random direction changed for every individual in each generation.

The next experiment compared these three repair directions: (i) operating repair from right to left, (ii) operating repair from left to right and (iii) operating repair in a random direction**.**

Table 2. The Random Direction GeneRepair Produced Best Results

	Min	Mean
Left-to-Right	471.44	519.67
Right-to-Left	483.36	529.27
Random	459.74	514.25

The results for this experiment are summarized in Table 2 above. Firstly, GeneRepair produces the best results when it proceeds in random and changing directions. The random strategy outperformed the two fixed direction strategies, on the best result generated and as an average across all runs of this experiment.

4.3.3 Fitness of Template

The final factor that we investigated was whether the fitness of the recorded ancestors had any impact on the goodness of the solutions generated. In the earlier experiments, at the end of each generation the genetic material of the fittest parent was recorded for each individual. This then formed part of the repair template for that individual.

In the next experiment, we explored the impact of recording a randomly selected parent for each individual. Thus for the randomly selected parent condition, the parent chosen to be moved into the repair genome was selected randomly, without any reference to the fitness of the two parents. It was expected that the superior fitness of the fittest condition would outperform the random parent conditions.

Table 3. Comparison of Fittest Ancestral Template with Random Ancestral Template

	Min	Mean
Random Parent	493.84	538
Fittest Parent	483.36	529.29
Random Grandparent	459.74	514.25
Fittest Grandparent	475.68	517.99

As shown above a random choice of ancestor proved to be superior to using the fittest of the two. This experiment also concretes the findings of the first experiment in that once again the grandparent was superior to the parent as a GeneRepair template. One explanation for the random ancestor from either generation outperforming the fittest ancestor may be deduced by examining the crossover technique used in this evolutionary strategy. The crossover is single point crossover where the point is chosen randomly for each individual. This means that the fittest ancestor does not necessarily have more impact on the individual than the other ancestor. We can theorize from this that because the fittest ancestor does not always have a larger impact on the individual it is not necessarily the best template to use for repair and a random template is more appropriate. The results above also confirm the results shown in Table 2 as the tour length of 459.74 which was achieved using a Random Grandparent repair template was found by conducting repair in a random direction

4.3.4 Penalty Points

We also examined the performance of the penalty points approach to enforce constraints, using the "death penalty" whereby invalid individuals are prevented from being used in crossover. The result of this experiment is shown in Table 4. As can be seen in Table 4 this approach produces significantly less-fit individuals that was produced by GeneRepair (Tables 1, 2 & 3).

Table 4. Death Penalty Approach

	Min	Mean
Death Penalty	1486.4	1584.94

4.3.5 Summary of Results

Each one of the repair directions described in Section 4.3.2 was tested for each of the inheritance template shown in Table 2 so there are in essence twelve different results to the experiment described in Section 4.3.3 rather than three.

The results of all twelve experiments are summarized in Fig. 3 below. The lines indicate the best solutions produced by each strategy across all runs of that strategy. The depicted results for each strategy were selected by choosing the best results at the end of the 500,000 generations. Each line indicates the best solution found thus far and as can be seen, this gradually converges towards the global optimal. (For this problem the known global optimal was 426. We were very pleased with the results of grandparent based repair – given that truncation selection was used. We expect even better results with Tournament selection and a much larger search space).

Best results are shown by the bottom line on Figure 3. This depicts the result for GeneRepair operating in a random direction using a randomly chosen grandparent as its repair template. This result is followed closely by the other GeneRepair techniques which use the grandparent as a repair template.

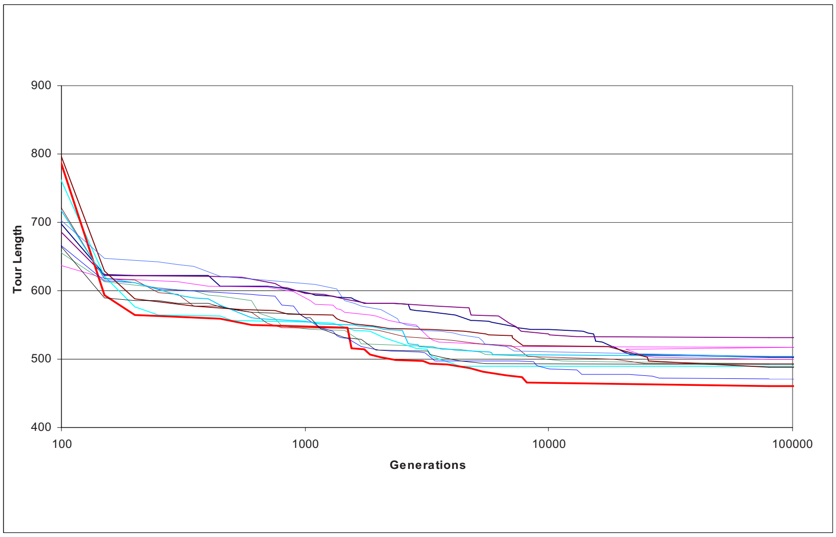


Fig. 3. Comparison of 12 different GeneRepair techniques

5 Conclusion

Evolutionary Optimization (EO) is an approach to optimization inspired by Darwin's idea of survival of the fittest. EO is a very are effective in exploring complex solution spaces, but are less suited to supporting validity constraints between the search parameters. This paper presents an approach to genetic repair that is inspired by the *Arabidopsis thaliana* plant. This plant is capable of making repairs to its own genes by making use of genetic information originating in the individuals grandparent. Controversially, this genetic information appears to skip the parent's generation. Our GeneRepair mechanism is inspired by an "archival DNA" explanation, though an alternative RNA based explanation exists. Errors in an individual plant's genes are repaired by comparison to the grandparents "template" DNA, which also serves to correct these errors.

We adapt this approach to genetic repair by applying it to a standard constrained optimization problem – the Traveling Salesman's Problem (TSP). This applied evolutionary optimization techniques, using unmodified crossover and mutation operators. An adjunction GeneRepair process ensured the validity of all solutions generated. This comparison found that GeneRepair based on a grandparent template produced better results than that of the parent based template. These results echoes recent advances in genetics, identifying non-Mendelian inheritance on the *Arabidopsis thaliana* plant [3]. A subsequent experiment indicates that archiving a randomly chosen parent produced better results than biasing the genetic archival process in favor of the fittest parent – and thus fittest grandparent. Our final experiment showed that the GeneRepair process produces best results when operating in random (and changing) directions. This approach outperformed both of the fixed direction strategies tested.

Not only do our results echo the controversial theory of Lolle *et al* [3], they also shed new light on this theory of non-Mendelian inheritance. First, that the repair seems to work best when it uses a randomly chosen grandparent and, secondly that repair should repair violations in a random order. Building upon Lolle's [3] results, these findings suggest a general approach to enforcing constraints on combinatorial optimization problems, opening up new possibilities for exploration.

Acknowledgement. We thank the SFI/HEA Irish Centre for High-End Computing (ICHEC) for their facilities and support.

References

[1] Coello Coello, C.: Theoretical and Numerical Constraint Handling Techniques in Evolutionary Algorithms: A Survey. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
[2] Kalyanmoy, D.: An Efficient Constraint Handling Method for Genetic Algorithms. Comp. Methods in Applied Mechanics & Engineering 186(2-4), 311–338 (2000)
[3] Lolle, S.J., Victor, J.L., Young, J.M., Pruitt, R.E.: Genome-wide non-mendelian inheritance of extra-genomic information in Arabidopsis. Nature 434, 505–509 (2005)
[4] Chaudhury, A.: Hothead healer and extragenomic information. Nature 437, E1-E2 (2005)
[5] Ray, A.: Plant genetics: RNA cache or genome trash? Nature 437, E1–E2 (2005)
[6] Michalewicz, Z., Schoenauer, M.: Evolutionary Algorithms for Constrained Parameter Optimization Problems. Evolutionary Computation 4(1), 1–32 (1996)
[7] Mitchell, G.G.: Evolutionary computation applied to Combinatorial Optimization Problems. PhD Thesis, Dublin City University, Dublin, Ireland (2007)
[8] Mitchell, G.G., O'Donoghue, D.P., Trenaman, A.: A New Operator for Efficient Evolutionary Solutions to the Traveling Salesman Problem. Applied Informatics, 0-88986-280-X, 771-774 (2000)
[9] Orvosh, D., Davis, L.D.: Shall We Repair? In: Proc. 5th Intl. Conf. on Genetic Algorithms (1993)

Improved Multilabel Classification with Neural Networks

Rafał Grodzicki[1], Jacek Mańdziuk[1], and Lipo Wang[2]

[1] Faculty of Mathematics and Information Science
Warsaw University of Technology
Plac Politechniki 1, 00-661 Warszawa, Poland
{R.Grodzicki,J.Mandziuk}@mini.pw.edu.pl
[2] School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, Nanyang Avenue, Singapore 639798
elpwang@ntu.edu.sg

Abstract. This paper considers the multilabel classification problem, which is a generalization of traditional two-class or multi-class classification problem. In multilabel classification a set of labels (categories) is given and each training instance is associated with a subset of this label-set. The task is to output the appropriate subset of labels (generally of unknown size) for a given, unknown testing instance. Some improvements to the existing neural network multilabel classification algorithm, named BP-MLL, are proposed here. The modifications concern the form of the global error function used in BP-MLL. The modified classification system is tested in the domain of functional genomics, on the yeast genome data set. Experimental results show that proposed modifications visibly improve the performance of the neural network based multilabel classifier. The results are statistically significant.

Keywords: multilabel, learning system, neural network, backpropagation, bioinformatics, functional genomics.

1 Introduction

Multilabel classification is a generalization of traditional two-class or multi-class classification. In both cases a finite set of labels (categories) is given, but unlike in the latter case, where the task is to associate each problem instance with one category, the multilabel classification associates each instance with a subset of the set of labels. In other words, a multilabel classifier transforms the domain of instances X to the power set of the set of labels 2^Y:

$$h: X \rightarrow 2^Y \tag{1}$$

where $X \subseteq R^d$ denotes the set of instances and $Y = \{0, 1, \ldots, Q\text{-}1\}$ represents the set of possible labels. In many practical situations the mulitilabel classification problem is converted to the problem of defining a function $f: X \times Y \rightarrow R$ such that for any $x_p \in X$

$$f(x_p, y_1) > f(x_p, y_2) \tag{2}$$

for all $y_1 \in Y_p$ and $y_2 \notin Y_p$. In other words instead of defining the classification of the form (1) it is sufficient to find function *f(.,.)* which provided higher outputs for the elements belonging to Y_p than for those not belonging to Y_p, where (x_p, Y_p) is a training (testing) instance.

Many real-world problems can be modeled by multilabel classification systems. The most popular application domains include text categorization [6], [7] and bioinformatics [3], in particular a functional genomics area. This latter problem is considered in this paper in order to verify the efficacy of proposed modifications.

Our work is inspired by Min-Ling Zhang and Zhi-Hua Zhou's paper [1], where a neural network based method is proposed as an approach to multilabel classification problem in the domains of functional genomics and text categorization. Up to our knowledge, the paper [1] is the first attempt to apply neural networks to multilabel classification. Previous approaches include using decision trees [4], [11] and kernel methods [3], [8]. In the case of single-label classification problem, current studies are based on kernel methods. However, experimental results presented in [1] indicate that in the case of multilabel classification, neural network based method [1] outperforms kernel method proposed in [3]. So we decided to consider neural network approach.

The paper is organized as follows: in the next section a brief description of the neural network based multilabel classifier introduced in [1] is presented together with some modifications to the global error function proposed in this paper. The most popular performance measures that are applied in this paper are also introduced. Experimental results in the domain of functional genomics are presented and discussed in Section 3. The last section is devoted to conclusions and possible future work.

2 Neural Networks in Multilabel Classification

The simplest approach to solve the multilabel classification problem is its decomposition into multiple set of classification problems – one for each label. This solution, however, has a significant disadvantage – it does not take into account dependencies between different categories. Hence a different approach need to be employed. One of the candidate algorithms is the well-known BackPropagation (BP) learning method [9], [10] which, after appropriate adaptation to the multilabel classification case, can be used to solve the problem. This idea was exploited in [1], where the algorithm named BP-MLL (Backpropagation for Multilabel Learning) was developed and experimentally verified.

2.1 A Brief Description of BP-MLL

BP-MLL is applied in [1] to a multilayer perceptron with sigmoidal neurons with one hidden layer and additional biases from the input and hidden layer. The size of the input layer is equal to the instance domain dimension (plus a bias neuron). The size of the output layer equals the number of labels (i.e. Q). Training is based on the classical BP algorithm, but in order to address the dependencies between labels, the new global error function of the following form is proposed:

$$E_1 = \sum_{p=1}^{m} \frac{\sum_{(r,s)\in Y_p \times \bar{Y}_p} e^{-\left(c_r^p - c_s^p\right)}}{|Y_p||\bar{Y}_p|} \tag{3}$$
$$h(x_p) = \{q \in Y : c_q(x_p) > t(x_p)\}, \quad c_q(x_p) = c_q^p$$

where m is the number of learning pairs, $Y_p \subseteq Y = \{0, 1, ..., Q-1\}$ is the set of labels associated with the *p-th* training instance, c_q^p (named *rank value*) is the actual output value of the *q-th* output neuron (corresponding to the *q-th* label), $\overline{Y_p}$ denotes the complementary set of Y_p (i.e. $\overline{Y_p} = Y \setminus Y_p$) and $h(x_p)$ denotes the set of labels attached to x_p by the network. Minimizing (3) tends to get higher output values by neurons corresponding to the labels belonging to Y_p than those not belonging to Y_p.

The next step to achieve multilabel classifier is determining the set of labels belonging to the input instance. This information can be retrieved from the neural network output values (rank values) by means of the threshold function which depends on the input vector. If the output neuron value is higher than the threshold value, then corresponding label belongs to the input instance. Otherwise, the label does not belong to the instance. More detailed description of BP-MLL can be found in [1].

2.2 Error Function Modifications

In this paper we propose some improvements of the error function used in [1]. The first introduced modification is integration of the threshold value into the error function used in BP-MLL. It results in the following form of the error function:

$$E_2 = \sum_{p=1}^{m} \frac{\sum_{(r,s)\in Y_p \times \bar{Y}_p} e^{-\left(c_r^p - c_s^p\right)} + \sum_{r\in Y_p} e^{-\left(c_r^p - c_Q^p\right)} + \sum_{s\in \bar{Y}_p} e^{-\left(c_Q^p - c_s^p\right)}}{|Y_p||\bar{Y}_p| + |Y_p| + |\bar{Y}_p|} \tag{4}$$
$$h(x_p) = \{q \in Y : c_q(x_p) > c_Q(x_p)\}, \quad c_q(x_p) = c_q^p$$

The last output neuron's value (c_Q^p) is interpreted as the threshold. The meaning of the remaining output neurons is the same as in case of using the BP-MLL method. Proposed solution allows to determine the threshold value by adaptation during neural network learning. Hence, unlike in the method described in [1], additional step devoted to definition of the threshold function is not required.

The above error function (4) can be further generalized (and the whole process becomes more autonomous) by introducing independent thresholds for different labels:

$$E_3 = \sum_{p=1}^{m} \frac{\sum_{(r,s)\in Y_p \times \bar{Y}_p} e^{-\left(c_{2r}^p - c_{2s}^p\right)} + \sum_{r\in Y_p} e^{-\left(c_{2r}^p - c_{2r+1}^p\right)} + \sum_{s\in \bar{Y}_p} e^{-\left(c_{2s+1}^p - c_{2s}^p\right)}}{|Y_p||\bar{Y}_p| + |Y_p| + |\bar{Y}_p|} \tag{5}$$
$$h(x_p) = \{q \in Y : c_{2q}(x_p) > c_{2q+1}(x_p)\}, \quad c_q(x_p) = c_q^p$$

In the case of equation (5) two output neurons (indexed by *2q* and *2q+1*) per each category *q* are considered. The first one of them (number *2q*) represents the output of the respective category (label) like in (3), and the other one (number *2q+1*) defines the respective threshold value for the *q-th* category.

Finally, in the error function comparisons between all the rank values of categories belonging to Y_p ($\overline{Y_p}$ resp.) and their respective threshold values can be taken into account, which leads to the following equations of the error function (6) and (7):

$$E_4 = \sum_{p=1}^{m} \frac{\sum_{(r,s)\in Y_p\times\overline{Y}_p} e^{-\left(c_{2r}^p - c_{2s}^p\right)} + \sum_{r\in Y_p}\sum_{t\in Y_p} e^{-\left(c_{2r}^p - c_{2t+1}^p\right)} + \sum_{s\in\overline{Y}_p}\sum_{t\in\overline{Y}_p} e^{-\left(c_{2t+1}^p - c_{2s}^p\right)}}{\left|Y_p\right|\left|\overline{Y}_p\right| + \left|Y_p\right|^2 + \left|\overline{Y}_p\right|^2} \tag{6}$$
$$h(x_p) = \{q \in Y : c_{2q}(x_p) > c_{2q+1}(x_p)\},\quad c_q(x_p) = c_q^p$$

$$E_5 = \sum_{p=1}^{m} \frac{\sum_{(r,s)\in Y_p\times\overline{Y}_p} \left(e^{-\left(c_{2r}^p - c_{2s}^p\right)} + e^{-\left(c_{2s+1}^p - c_{2r+1}^p\right)}\right) + \sum_{r\in Y_p}\sum_{t\in Y_p} e^{-\left(c_{2r}^p - c_{2t+1}^p\right)} + \sum_{s\in\overline{Y}_p}\sum_{t\in\overline{Y}_p} e^{-\left(c_{2t+1}^p - c_{2s}^p\right)}}{2\left|Y_p\right|\left|\overline{Y}_p\right| + \left|Y_p\right|^2 + \left|\overline{Y}_p\right|^2} \tag{7}$$
$$h(x_p) = \{q \in Y : c_{2q}(x_p) > c_{2q+1}(x_p)\},\quad c_q(x_p) = c_q^p$$

Note that (6) and (7) differ from (5), where each rank value is compared only with one threshold assigned to it. In (6) and (7) each rank value of category belonging to Y_p ($\overline{Y_p}$ resp.) is compared with each threshold value of category belonging to Y_p ($\overline{Y_p}$ resp.) Moreover, (7) extends (6) by also considering differences between threshold values (cf. the first terms in the numerators of both equations). In effect minimization of (7) results in lower threshold values corresponding to labels belonging to Y_p , for *p = 1, …, m,* than to those not belonging to this set.

2.3 Evaluation Metrics

Before presentation of experimental results let us briefly introduce the most popular error measures used in multilabel classification domain [1], [3]. The three of them, namely the *Hamming loss*, the *one-error* and the *ranking loss* are considered in this paper.

The *Hamming loss* measure (8) indicates the frequency (with respect to the size of the testing set *K*) of incorrect classification (the instance is classified as associated with particular label when it is actually not the case or *vice-versa* the instance is not classified as associated with this label in case it should be).

$$hloss(h) = \frac{1}{K}\sum_{p=1}^{K}\frac{1}{Q}\left|h(x_p)\Delta Y_p\right|, \tag{8}$$
$$h(x_p)\Delta Y_p = \left(h(x_p)\cup Y_p\right)\setminus\left(h(x_p)\cap Y_p\right)$$

The *one-error* measure (9) points out how often the label with the highest rank value (the top-one) does not belong to Y_p. Function *f* in (9) is defined by equation (2).

$$one-error(f)=\frac{1}{K}\sum_{p=1}^{K}\left[\left(\arg\max_{y\in Y} f(x_p,y)\right)\notin Y_p\right] \tag{9}$$

The third error measure, the *ranking loss* (10) indicates how often the label belonging to Y_p has got lower or equal rank value than the one not belonging to Y_p (which is not the expected outcome).

$$rloss(f)=\frac{1}{K}\sum_{p=1}^{K}\frac{1}{|Y_p||\overline{Y}_p|}\left|\left\{(y_1,y_2)\in Y_p\times\overline{Y}_p : f(x_p,y_1)\leq f(x_p,y_2)\right\}\right| \tag{10}$$

Ranking loss and *one-error* denote the function *f* defined in (2) and *Hamming loss* denotes the function *h* defined in (1). Both *Hamming loss* and *ranking loss* consider the frequencies of incorrect output values (in the case of *Hamming loss* incorrect label assignment and in the case of *ranking loss* incorrect order of rank values). Both measures well address multilabel classification characteristics. *One-error* takes into account only the label with the highest rank value and ignores other labels. *One-error* does not measure general performance of multilabel classifier but is specialized for penalizing classifiers which frequently give the highest rank value to the labels not belonging to Y_p.

3 Application to Functional Genomics

Our modifications to the BP-MLL method were tested against real-world multilabel classification problem in the domain of functional genomics [1], [3], [4], [5]. The goal is to determine (various) functions of genes based on biological data such as gene expression levels [5] (from DNA micro arrays), sequence data (sequences of nucleotides or amino acids) or phylogenetic profiles [5].

3.1 Yeast Genome Data Set and Learning Parameters

In particular in our experiments a data set [2] dealing with yeast genome was considered. This set was also used by other researchers, e.g. [1], [3]. It contains 2417 genes associated with functional classes. Every gene is described by 103-dimensional vector consisting of the information about phylogenetic profile and gene expression levels. This vector forms the neural network input. Each input vector is associated with a subset of the set of 14 possible functional classes. In average, each example (gene) is associated with 4.24 ± 1.57 labels.

During training process the learning rate was set to 0.05. In order to avoid overfitting, similarly to [1] a weight decay (equal to 0.5) was introduced. Due to the relatively small size of the data set, tenfold cross-validation was applied. Training was performed separately on each of the five multilabel classifiers – one for each global error function (3) – (7).

The size of a hidden layer was equal to 20 and 40 neurons, respectively for the classifiers with error functions (3), (4) and (5), (6), (7). The number of training epochs was equal to 100. In the case of (3) the threshold function (*t*) was set to be the zero constant function.

3.2 Experimental Results

Five experiments, each based on tenfold cross validation, were performed for each classifier. This resulted in 50 evaluations of each of the tested classifiers for each of the three error measures (Hamming loss, one-error and ranking loss). Table 1 presents means and standard deviations of those evaluations. Results of statistical tests (t-test at 5 percent significance level) are shown in Table 2.

Table 1. Means and standard deviations of considered multilabel classifiers evaluations

Error function	Hamming loss		One-error		Ranking loss	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
(3)	0,2754	0,0188	0,2324	0,0292	0,1729	0,0150
(4)	0,2005	0,0071	0,2311	0,0255	0,1705	0,0102
(5)	0,2023	0,0094	0,2351	0,0215	0,1721	0,0107
(6)	0,1988	0,0094	0,2252	0,0230	0,1659	0,0120
(7)	0,1987	0,0089	0,2247	0,0242	0,1657	0,0117

Table 2. Statistical tests results (t-test at 5 percent significance level). The results below the 5 percent level are presented in boldface.

Test	Hamming loss p-value	One-error p-value	Ranking loss p-value
(3) vs. (4)	**0**	0,8202	0,3569
(3) vs. (5)	**0**	0,5950	0,7786
(3) vs. (6)	**0**	0,1778	**0,0116**
(3) vs. (7)	**0**	0,1578	**0,0089**
(4) vs. (5)	0,2787	0,4010	0,4335
(4) vs. (6)	0,2981	0,2289	**0,0410**
(4) vs. (7)	0,2594	0,2026	**0,0310**

The results of experiments presented in both tables allow to make some performance comparisons between various neural network multilabel classifiers taken into account. Considering the Hamming loss shows that all modified classifiers (i.e. (4), (5), (6) and (7)) are significantly better than the original one (3). Moreover, there are no statistically significant differences in the results accomplished by classifier (4) vs. (5), (6) or (7). One-error performance measure does not permit to make any conclusions about potential differences between multilabel classifiers in question. They are statistically comparable. This can be caused by the characteristics of one-error measure which only considers some details of classifier and does not address multilabel classifier performance in general. Finally, classifiers (6) and (7) outperform (3) and (4), with statistical significance when ranking loss measure is considered.

Table 3 presents comparison of BP-MLL with other approaches to multilabel classification (decision tree based method ADTBOOST.MH [11] and kernel method RANK-SVM [3]) on the Yeast Genome Data Set considered in this paper. This

Table 3. Mean values and standard deviations of considered multilabel classifiers evaluations for three approaches to multilabel classification (BP-MLL, ADTBOOST.MH and RANK-SVM)

Error function	Hamming loss		One-error		Ranking loss	
	Mean	Std.Dev.	Mean	Std.Dev.	Mean	Std.Dev.
BP-MLL	0,206	0,011	0,233	0,034	0,171	0,015
ADTBOOST.MH	0,207	0,010	0,244	0,035	-	-
RANK-SVM	0,207	0,013	0,243	0,039	0,195	0,021

comparison was made by authors of [1] and shows that BP-MLL outperforms both ADTBOOST.MH and RANK-SVM.

4 Conclusions and Future Work

Multilabel classification problem generalizes traditional two-class or multi-class classification since each instance in the training/testing set is associated with several (usually more than one) classes (labels). The problem is not easy to solve also because the size of the label-set associate with particular unseen instance is generally unknown. Various approaches to tackle this problem were presented in the literature, but – up to our knowledge – there has been only one attempt to apply a neural network for solving this task [1]. In this paper a few modifications of the global error function proposed in [1] are presented and experimentally evaluated. Generally, all of them improve performance of the multilabel neural classifier. The improvement – in case of the two most elaborate functions, i.e. (6) and (7) is noticeable and statistically significant. Overall, including the threshold values into the error function and considering differences between the rank values and the thresholds proved to be a promising direction for improvement of the multilabel classifier performance.

Currently, we are focused on performing more tests (especially with other sizes of hidden layer) and on other data sets in order to further verify the efficacy of proposed modifications.

References

1. Zhang, M.L., Zhou, Z.H.: Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. IEEE Transactions on Knowledge and Data Engineering 18(10), 1338–1351 (2006)
2. LIBSUM Data: Multilabel Classification, `http://www.csie.ntu.tw/~cjlin/libsumtools/datasets/multilabel.html#yeast`
3. Elissef, A., Weston, J.: A Kernel Method for Multi-Labelled Classification. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14, pp. 681–687 (2002)
4. Clare, A., King, R.D.: Knowledge Discovery in Multi-Label Phenotype Data. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001)

5. Pavlidis, P., Weston, J., Cai, J., Grundy, W.N.: Combining Microarray Expression Data and Phylogenetic Profiles to Learn Functional Categories using Support Vector Machines. In: 5th Annual International Conference Computational Molecular Biology (RECOMB 2001), pp. 242–248 (2001)
6. McCallum, A.: Multi-Label Text Classification with a Mixture Model Trained by EM. In: Working Notes Am. Assoc. Artificial Intelligence Workshop Text Learning (AAAI 1999) (1999)
7. Schapire, R.E., Singer, Y.: BoosTexter: A Boosting-Based System for Text Categorization. Machine Learning 39(2/3), 135–168 (2000)
8. Kazawa, H., Izumitani, T., Taira, H., Maeda, E.: Maximal Margin Labeling for Multi-Topic Text Categorization. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17, pp. 649–656 (2005)
9. Werbos, P.J.: Beyond Regression: New Tools for Prediction and Anlysis in the Behavioral Sciences. PhD thesis, Harvard University (1974)
10. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Propagation. In: Rumelhart, D.E., McClelland, J.L. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, pp. 318–362 (1986)
11. Comite, F.D., Gilleron, R., Tommasi, M.: Learning Multi-Label Alternating Decision Tree from Texts and Data. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNCS, vol. 2734, pp. 35–49. Springer, Heidelberg (2003)

Enhancing Efficiency of Hierarchical BOA Via Distance-Based Model Restrictions

Mark Hauschild and Martin Pelikan

Missouri Estimation of Distribution Algorithms Laboratory, 320 CCB; University of Missouri in St. Louis; One University Blvd., St. Louis, MO 63121
mwh308@umsl.edu pelikan@cs.umsl.edu

Abstract. This paper analyzes the effects of restricting probabilistic models in the hierarchical Bayesian optimization algorithm (hBOA) by defining a distance metric over variables and disallowing dependencies between variables at distances greater than a given threshold. We argue that by using prior problem-specific knowledge, it is often possible to develop a distance metric that closely corresponds to the strength of interactions between variables. This distance metric can then be used to speed up model building in hBOA. Three test problems are considered: 3D Ising spin glasses, random additively decomposable problems, and the minimum vertex cover.

1 Introduction

The hierarchical Bayesian Optimization Algorithm (hBOA) [1,2] has been shown to solve a large range of problems scalably and robustly. However, being able to solve a problem in low-order polynomial time is not always enough. As problem size and difficulty increases, the computational resources necessary could still make the problem intractable in practice. That is why it is important to design efficiency enhancement techniques [3,4,5,6,7] which can further improve the efficiency of hBOA and other estimation of distribution algorithms (EDAs) [8,9,10,2].

One approach to speeding up EDAs is to use prior knowledge to restrict or bias model building [11,12,13,14]. One way to do this is to define a distance metric on variables such that the variables that are close to each other with respect to this metric are expected to influence each other more strongly. If this metric accurately reflects the problem structure, one can use it to speed up hBOA and other EDAs with complex models by disallowing dependencies between variables at a distance above a given threshold [12,13,14]. This should improve algorithm performance by simplifying model building and increasing model accuracy.

In this paper we study the effects of such model restrictions on three important classes of problems: 3D spin glasses, random additively decomposable problems, and minimum-vertex cover. First, a distance metric is constructed for each problem based on the structure of the objective function. The metric is then used to restrict models in hBOA, and hBOA performance is tested for various values of the threshold. The results show that when hBOA is given a reasonable threshold, model restrictions lead to substantial speedups.

The paper is organized as follows. Section 2 discusses prior work on biasing model building in EDAs. Section 3 outlines hBOA. Section 4 discusses the design of good distance metrics and the expected benefits of using an appropriate distance metric to restrict model structure. Section 5 describes test problems and the distance metrics for each problem. Section 6 presents experimental results. Finally, section 7 summarizes and concludes the paper.

2 Previous Work on Biasing Model Building in EDAs

There are two main approaches to biasing model building in EDAs: (1) Impose *soft restrictions* by biasing the scoring metric to prefer models that closely correspond to the problem structure [11,15] or (2) impose *hard restrictions* by strictly disallowing some dependencies [12,13,14,15].

Schwarz & Ocenasek [11] proposed the use of prior probabilities of competing network structures in BOA to bias model building toward models that closely correspond to the problem structure in graph bi-partitioning. Edges between variables connected in the underlying graph were thus given preference, but no edges were strictly disallowed.

Mühlenbein & Mahnig [12] also considered graph bi-partitioning but they used a hard restriction to only allow connections between nodes connected in the underlying graph. Baluja [13] proposed the use of the same hard restriction in the dependency-tree EDA on graph coloring. Santana [14] used the same hard restriction to speed up model building of dependency trees on a protein design problem. Hauschild et al. [15] proposed the use of both soft restrictions based on prior models on problems of the same structure as well as hard restrictions based on a distance metric for hBOA on the 2D spin glass and MAXSAT. This paper extends the work of Hauschild et al. [15] on hard restrictions based on a distance-metric by considering other important classes of problems, including two NP-complete problems (3D spin glass and minimum vertex cover).

3 Hierarchical BOA (hBOA)

Estimation of distribution algorithms (EDAs) [8,9,16,2,17] replace standard crossover and mutation operators of genetic algorithms by building an explicit probabilistic model of selected solutions and sampling the built model to generate new candidate solutions. hBOA is an EDA that uses Bayesian networks as probabilistic models and incorporates restricted tournament replacement [18] for effective diversity maintenance.

hBOA evolves a population of candidate solutions represented by fixed-length strings over a finite alphabet (e.g., binary strings). The initial population is generated at random according to the uniform distribution over all potential solutions. Each iteration (generation) starts by selecting promising solutions from the current population using any standard selection method of genetic and evolutionary algorithms. In this paper we use truncation selection with threshold

$\tau = 50\%$. Next, hBOA builds a Bayesian network [19,20] with local structures [21,22] as a model of the selected solutions using a greedy algorithm. Learning the model structure is typically the most challenging task of model building [2]. New solutions are generated by sampling the built network. These are then incorporated into the original population using restricted tournament replacement (RTR) [18], which ensures effective diversity maintenance. We set the window size $w = \min\{n, N/20\}$, where n is the number of decision variables and N is the population size [2]. The next iteration is then executed unless some predefined termination criteria are met. For more details, see refs. [21,1,2,22].

Since a simple deterministic hill climber (DHC) was shown to lead to substantial speedups of hBOA on both the 3D spin glass [2] and the minimum vertex cover [23], we incorporated DHC into hBOA for these two problems. DHC takes a candidate solution and performs one-bit changes on it that lead to the maximum improvement in fitness and DHC is terminated when no single-bit flip leads to improvement.

4 Distance-Based Model Restriction

To improve model building in hBOA, only necessary dependencies should be considered. This may significantly reduce the space of potential model structures, improving the speed and accuracy of model building. Doing this is often not easy in practice, as for many problems it is difficult to identify necessary dependencies. Nonetheless, it is often possible to provide a soft measure that ranks edges according to their expected importance based on prior problem-specific knowledge. The likelihood of certain dependencies can be estimated for example from an explicit or implicit distance metric between problem variables. In many problems, such a distance metric is relatively easy to design from the structure of the objective function or previous runs of an EDA on similar problem instances.

For example, for 2D spin glasses, Hauschild et al. [24] showed that while in general it is not easy to decide what dependencies are unnecessary, dependencies are more likely to connect spins located close to each other with respect to the shortest path between these spins in the underlying 2D lattice. This fact is explored by Hauschild et al. [15], who examined the effects of specifying a distance threshold and disallowing dependencies between spins at a distance above the threshold (for example, only allowing dependencies between spins that were of distance 2 or less from each other). A similar distance metric was proposed and tested for MAXSAT. The results showed that with an appropriate threshold, distance-based restrictions lead to substantial speedups on both problems. This paper considers similar restrictions on three additional classes of difficult problems, two of which are NP-complete.

We expect two main benefits from restricting model structure in this way. By disallowing some dependencies, model building should become significantly faster because there are fewer dependencies to examine. Additionally, by disallowing unlikely dependencies, the model should contain fewer spurious dependencies, leading to improved model accuracy. These two factors should lead to significant speedup of hBOA as is also confirmed in section 6.

The following section describes the test problems considered in this paper and the distance metrics used to restrict models for all problems.

5 Test Problems

5.1 3D Ising Spin Glass with $\pm J$ Couplings

An Ising spin glass is typically arranged on a regular 2D or 3D grid where each node i corresponds to a spin s_i and each edge $\langle i, j\rangle$ corresponds to a coupling between two spins s_i and s_j. For the classical Ising model, each spin s_i can be in one of two states: $s_i = +1$ or $s_i = -1$. Each edge has a real value $J_{i,j}$ associated with it that defines the relationship between the two connected spins. Periodic boundary conditions are used that introduce a coupling between the first and the last elements along each dimension.

Here the task is to find spin configurations C that minimize the energy for a given set of coupling constants $J_{i,j}$, defined as

$$E(C) = \sum_{\langle i,j\rangle} -s_i J_{i,j} s_j \ , \tag{1}$$

where the sum runs over all couplings $\langle i, j\rangle$. The minimum-energy configurations are called ground states. To represent spin configurations, we use binary strings of length n where ith bit defines the value of the ith spin (-1 is encoded by 0, +1 is encoded by 1). We consider random instances of the 3D $\pm J$ spin glass, where each coupling constant is set randomly to $+1$ or -1 with equal probability. The problem of finding ground states of 3D spin glasses is NP-complete [25] and, due to its complex landscape, it poses a challenge for most optimization algorithms.

Based on the results on 2D spin glasses [24,15], we define the distance between two spins as the shortest path between these spins in the underlying 3D grid.

5.2 Random Additively Decomposable Problems

Random additively decomposable problems (rADPs) [26,27] are a class of test problems developed to test performance of evolutionary algorithms on broad classes of decomposable problems. The input string in rADPs is partitioned into subsets of bits, with the overall fitness being the sum of the subfunctions applied to all the subsets. We denote the order of rADPs by k; that is, each subproblem contains k bits. To ensure that rADP instances are solvable in polynomial time, the subproblems are located in contiguous blocks of k bits and the overlap is specified by a parameter o which denotes the number of bits shared by neighbor subproblems. The fitness for each subproblem is given by a table of 2^k values which are generated randomly from the uniform distribution over $[0, 1)$. To verify the global optimum, the dynamic-programming algorithm [28] is used. Instances of rADPs vary in difficulty due to the differences in subfunction difficulty and the amount of overlap.

Intuitively, bits located in the same subproblem are likely to influence each more strongly than bits located in different subproblems regardless of the overlap. More generally, in the presence of overlap, we can expect that the greater the distance between the subproblems containing two bits, the more weakly the two bits influence each other [24]. To design a distance metric that captures this fact, we create a graph that connects pairs of bits located in the same subproblem and define the distance between each such pair of bits as 1. The distance between any two bits is then computed as the shortest path between these bits in this graph.

5.3 Minimum Vertex Cover

The minimum vertex cover (MVC) of an undirected graph G is the smallest subset of nodes in G such that for every edge G, at least one of the two nodes this edge connects is in this subset. MVC is interesting because it is NP-complete [29] and is closely related to other hard graph problems. In this paper we consider MVC for random instances of $G(n, m)$ graphs [30,31]. $G(n, m)$ consists of graphs with n vertices and m edges such that $m = nc$, where $c > 0$ is a constant. To represent subsets of nodes in hBOA, we use n-bit binary strings where the ith bit is 1 if and only if the ith node is selected. A repair operator is used to ensure that each solution corresponds to a valid graph cover [23] and the fitness of each cover is defined as the number of nodes *not* contained in this cover.

Intuitively, bits corresponding to the nodes located closer in the underlying graph can be expected to influence each other more strongly. So we define the distance between bits as the minimum number of edges on a path between these vertices. Pairs of vertices in unconnected components are assigned distance n.

6 Experiments

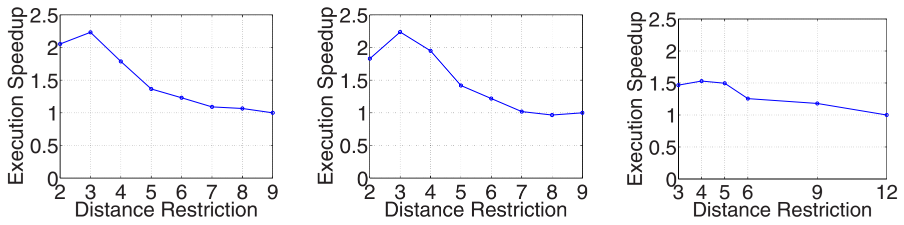
For all problem instances, bisection [32,2] was used to determine the minimum population size to ensure convergence to the global optimum in 5 out of 5 independent runs, with the results averaged over the 5 runs. The number of generations was upper bounded according to hBOA scalability theory [33] by n where n is the number of bits in the problem. Each run of hBOA is terminated when the global optimum has been found (success) or when the upper bound on the number of generations has been reached without discovering the global optimum (failure).

Various values of the distance threshold parameter are considered, from the maximum observed distance to the minimum distance required to solve all problem instances in a reasonable time. The results for certain thresholds are excluded since the restriction was too severe, requiring large population sizes, $N \geq 10^5$.

In the following sections, some of our graphs use the term *Reduction Factor*. This is the factor by which the numbers have been decreased from the base case.

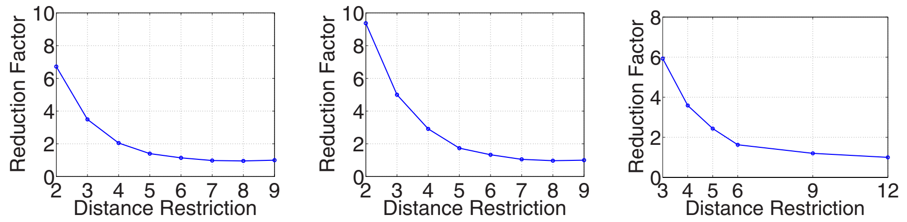
6.1 Results on 3D Spin Glasses

To examine the speedups obtained with distance-based model restrictions in 3D spin glasses, we considered three problem sizes: $6 \times 6 \times 6$, $7 \times 7 \times 7$ and $8 \times 8 \times 8$.



(a) $6 \times 6 \times 6$ (b) $7 \times 7 \times 7$ (c) $8 \times 8 \times 8$

Fig. 1. Execution time speedup by distance restriction on 3D spin glass



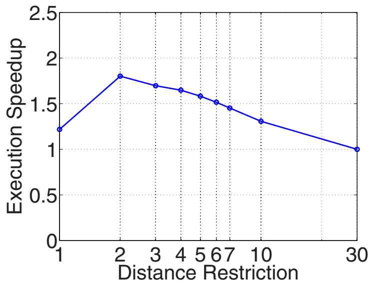
(a) $6 \times 6 \times 6$ (b) $7 \times 7 \times 7$ (c) $8 \times 8 \times 8$

Fig. 2. Reduction in the number of bits examined in model building on 3D spin glass

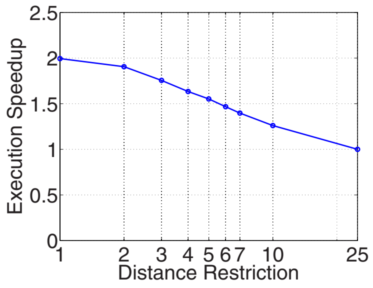
Since spin glass instances vary in difficulty, we considered 1000 different instances for the two smaller sizes and 100 different instances for the largest size.

Figure 1 shows the execution-time speedup for various distance thresholds for all three problem sizes. The results show that the model restriction yields speedups of 1.5 to 2.2. We also see that as the problem becomes larger, dependencies at larger distances are expected to be important. For the two smaller problems, the best threshold is 3, whereas for the largest problem the best thresholds are 4 and 5. In the two smaller problems, distance of at least 2 must be considered for efficient performance, whereas for the largest problem we must consider distances of at least 3. The results also show that while the best speedups obtained for the two smaller problems are almost identical, the best speedup obtained on the largest instance is smaller. The reason for this may be that only 100 problem instances could be considered for the largest problem due to its high complexity, and the results are affected by the specific selection of instances.

Figure 2 shows the reduction in the number of bits examined during the entire model-building procedure. The results show a dramatic reduction in the number of bits examined, which decreases monotonically with the threshold. Nonetheless, while reducing the number of bits examined, more severe model restrictions can also be expected to lead to an increased time complexity of fitness evaluation due to the larger population sizes required to achieve reliable performance.

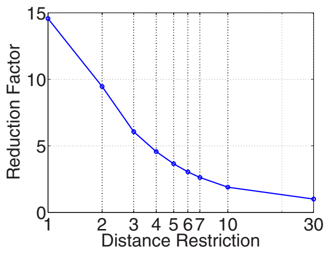


(a) $n = 92, k = 5, o = 2$

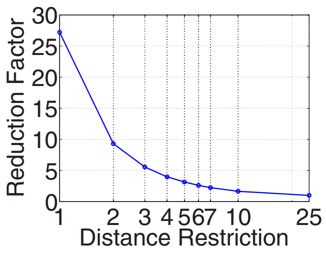


(b) $n = 101, k = 5, o = 1$

Fig. 3. Execution speedup by distance restriction on rADPs



(a) $n = 92, k = 5, o = 2$



(b) $n = 101, k = 5, o = 1$

Fig. 4. Reduction in the number of bits examined by distance restriction on rADPs

6.2 Results on rADPs

To examine distance-based model restrictions on rADPs, we considered two different combinations of values of n and o. For the heavier overlap of $o = 2$, we considered the problem of 92 bits. On the other hand, for overlap $o = 1$, we considered instances of $n = 101$ bits. In both cases, we set $k = 5$. 1000 random problem instances are tested for each problem size.

Figure 3a shows the execution time speedup for various distance thresholds on rADPs of $n = 92$ with $o = 2$. We see that the optimal speedup is obtained when we allow dependencies only between bits in the same subproblem or those that are part of overlapping subproblems. On the other hand, figure 3b shows that for $n = 101$ and $o = 1$, the best speedup is obtained when we allow only dependencies between the bits in the same subproblem. Therefore, it is clear that stronger overlap leads to the need of considering dependencies between bits at greater distances; on the other hand, weaker overlap allows more severe model restrictions.

Figure 4 shows the reduction in the number of bits examined during the entire model-building procedure for both cases. We see that the results for both cases are very similar. Similarly as for the 3D spin glass, the number of bits examined decreases with stronger restrictions.

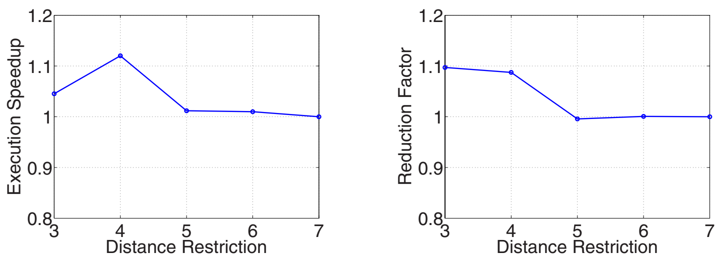


Fig. 5. Execution time speedup and reduction in the number of bits examined for MVC

Table 1. Percentage of pairs of nodes with different distance thresholds for random instances of the $G(n,m)$ model with $n = 100$ and $m = 400$

dist	1	2	3	4	5	6	none
total	1200000	8388911	26212460	8883471	137371	580	151960
prob	2.7%	21.3%	79.6%	99.4%	99.7%	99.7%	100%

6.3 Results on Minimum Vertex Cover

To examine the distance-based model restrictions on MVC, we considered 1000 random instances of the $G(n,m)$ model with $n = 300$ nodes and $m = 4n$.

Figure 5 (left-hand side) shows the execution-time speedup for various distance thresholds on MVC. The speedups obtained are much smaller than those obtained for the other two test problems. Even in the best case (threshold of 4) we only see slightly more than a 10% improvement. While this is somewhat surprising, the reason can be explained by the obtained reduction of the number of bits examined, which is shown in Figure 5 (right-hand side). Unlike in the previous experiments, the reduction in the number of bits examined reaches only a factor of 1.1 even for the most severe distance restrictions. We also see that there is a sharp drop in the number of bits examined between the thresholds of 4 and 5.

To examine the MVC results in more depth, we looked at the distribution of distances over all 1000 problem instances, which is shown in table 1. These results show that for thresholds of 5 or more, almost all pairs of nodes are allowed to be connected, which explains why the reduction in the number of bits examined is negligible. Furthermore, the results show that when decreasing the thresholds of 1 and 2 leads to a dramatic reduction in the number of possible dependencies, explaining poor performance of hBOA with such severe model restrictions.

7 Summary and Conclusions

This paper analyzed the speedups obtained in hBOA by defining a distance metric over problem variables and strictly disallowing dependencies between variables at a distance greater than a given threshold. Three test problems were considered: 3D spin glass, random additively decomposable problems (rADPs), and minimum vertex cover (MVC). It was shown that substantial speedups of 1.5 to 2.2 can be obtained for the 3D spin glass and rADPs, whereas only small speedups of about 1.1 can be obtained for MVC.

While we only looked at three problems, a distance metric can be defined for many other problems, including graph coloring, atomic cluster optimization, and the quadratic assignment problem. An important direction for future research is to extend the proposed techniques to other important classes of problems. Furthermore, since the benefits of using distance-based model restrictions depend on the distance threshold, an important direction for future work is to develop automated methods for choosing an appropriate threshold or to eliminate the threshold altogether using soft distance-based restrictions. Finally, similar ideas might be applied to other EDAs based on multivariate models.

An important thing to remember is that combining efficiency enhancement techniques can lead to multiplicative speedups [34,7]. That means that even a moderate speedup of 1.5 or 2 can significantly contribute to the overall efficiency.

Acknowledgments

This project was sponsored by the National Science Foundation, NSF, under CAREER grant ECS-0547013, by the Air Force Office of Scientific Research, AFOSR, under grant FA9550-06-1-0096, and by the Univ. of Missouri–St. Louis, UMSL, through the High Performance Computing Collaboratory sponsored by ITS, and the Research Award and Research Board programs. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, AFOSR, or the U.S. Government. Some experiments were done using the hBOA software developed by M. Pelikan and D. E. Goldberg at the Univ. of Illinois at Urbana-Champaign and most experiments were performed on the Beowulf cluster maintained by ITS at UMSL.

References

1. Pelikan, M., Goldberg, D.E.: Escaping hierarchical traps with competent genetic algorithms. In: Genetic and Evolutionary Computation Conf (GECCO 2001), pp. 511–518 (2001)
2. Pelikan, M.: Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms. Springer, Heidelberg (2005)
3. Sastry, K., Goldberg, D.E., Pelikan, M.: Don't evaluate, inherit. In: Genetic and Evolutionary Computation Conf .(GECCO 2001), pp. 551–558 (2001)
4. Pelikan, M., Sastry, K.: Fitness inheritance in the Bayesian optimization algorithm. In: Genetic and Evolutionary Computation Conf (GECCO 2004), vol. 2, pp. 48–59 (2004)

5. Lima, C.F., Pelikan, M., Sastry, K., Butz, M.V., Goldberg, D.E., Lobo, F.G.: Substructural neighborhoods for local search in the Bayesian optimization algorithm. Parallel Problem Solving from Nature, 232–241 (2006)
6. Pelikan, M., Sastry, K., Goldberg, D.E.: Sporadic model building for efficiency enhancement of hierarchical BOA. In: Genetic and Evolutionary Computation Conf (GECCO 2006), pp. 405–412 (2006)
7. Sastry, K., Pelikan, M., Goldberg, D.E.: Efficiency enhancement of estimation of distribution algorithms. In: Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.) Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, pp. 161–185. Springer, Heidelberg (2006)
8. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA (1994)
9. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. Parallel Problem Solving from Nature, 178–187 (1996)
10. Larranaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Springer, Heidelberg (2001)
11. Schwarz, J., Ocenasek, J.: A problem-knowledge based evolutionary algorithm KBOA for hypergraph partitioning, Personal communication (2000)
12. Mühlenbein, H., Mahnig, T.: Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. International Journal on Approximate Reasoning 31(3), 157–192 (2002)
13. Baluja, S.: Incorporating a priori knowledge in probabilistic-model based optimization. In: Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.) Scalable optimization via probabilistic modeling: From algorithms to applications, pp. 205–219. Springer, Heidelberg (2006)
14. Santana, R., Larrañaga, P., Lozano, J.A.: The role of a priori information in the minimization of contact potentials by means of estimation of distribution algorithms. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 247–257. Springer, Heidelberg (2007)
15. Hauschild, M., Pelikan, M., Sastry, K., Goldberg, D.E.: Analyzing probabilistic models in hierarchical boa on traps and spin glasses. In: Genetic and Evolutionary Computation Conference (GECCO 2008), vol. I, pp. 523–530 (2008)
16. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Boston (2002)
17. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications 21(1), 5–20 (2002)
18. Harik, G.R.: Finding multimodal solutions using restricted tournament selection. In: International Conference on Genetic Algorithms (ICGA 1995), pp. 24–31 (1995)
19. Howard, R.A., Matheson, J.E.: Influence diagrams. In: Howard, R.A., Matheson, J.E. (eds.) Readings on the principles and applications of decision analysis, vol. II, pp. 721–762. Strategic Decisions Group, Menlo Park (1981)
20. Pearl, J.: Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann, San Mateo (1988)
21. Chickering, D.M., Heckerman, D., Meek, C.: A Bayesian approach to learning Bayesian networks with local structure. Technical Report MSR-TR-97-07, Microsoft Research, Redmond, WA (1997)
22. Friedman, N., Goldszmidt, M.: Learning Bayesian networks with local structure. In: Jordan, M.I. (ed.) Graphical models, pp. 421–459. MIT Press, Cambridge (1999)

23. Pelikan, M., Kalapala, R., Hartmann, A.K.: Hybrid evolutionary algorithms on minimum vertex cover for random graphs. In: Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 547–554 (2007)
24. Hauschild, M., Pelikan, M., Lima, C., Sastry, K.: Analyzing probabilistic models in hierarchical boa on traps and spin glasses. In: Genetic and Evolutionary Computation Conference (GECCO 2007), vol. I, pp. 523–530 (2007)
25. Barahona, F.: On the computational complexity of Ising spin glass models. Journal of Physics A: Mathematical, Nuclear and General 15(10), 3241–3253 (1982)
26. Pelikan, M., Sastry, K., Butz, M.V., Goldberg, D.E.: Performance of evolutionary algorithms on random decomposable problems. Parallel Problem Solving from Nature (PPSN IX), 788–797 (2006)
27. Sastry, K., Pelikan, M., Goldberg, D.E.: Empirical analysis of ideal recombination on random decomposable problems. In: Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 1388–1395 (2007)
28. Pelikan, M., Sastry, K., Butz, M.V., Goldberg, D.E.: Hierarchical BOA on random decomposable problems. MEDAL Report No. 2006001, Missouri Estimation of Distribution Algorithms Laboratory, Univ. of Missouri–St. Louis (2006)
29. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)
30. Bollobas, B.: Random Graphs. Cambridge University Press, Cambridge (2001)
31. Weigt, M., Hartmann, A.K.: Minimal vertex covers on finite-connectivity random graphs - a hard-sphere lattice-gas picture. Physical Review E 63, 056127 (2001)
32. Sastry, K.: Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, Department of General Engineering, Urbana, IL (2001)
33. Pelikan, M., Sastry, K., Goldberg, D.E.: Scalability of the Bayesian optimization algorithm. International Journal of Approximate Reasoning 31(3), 221–258 (2002)
34. Goldberg, D.E.: The design of innovation: Lessons from and for competent genetic algorithms. Kluwer, Dordrecht (2002)

Evolution Strategies for Direct Policy Search

Verena Heidrich-Meisner and Christian Igel

Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany
{Verena.Heidrich-Meisner,Christian.Igel}@neuroinformatik.rub.de

Abstract. The covariance matrix adaptation evolution strategy (CMA-ES) is suggested for solving problems described by Markov decision processes. The algorithm is compared with a state-of-the-art policy gradient method and stochastic search on the double cart-pole balancing task using linear policies. The CMA-ES proves to be much more robust than the gradient-based approach in this scenario.

1 Introduction

Reinforcement learning (RL) aims at maximizing accumulated reward over time by improving a behavioral policy mapping states to actions. The learning is based on interaction with the environment, where perceived transitions between states and scalar reward signals, which may be sparse, noisy, and/or delayed, drive the adaptation [1,2,3]. Various evolutionary algorithms (EAs) have been successfully applied to RL problems (see, e.g., [4,5,6,7]) and performed well in comparison with alternative approaches (see [8,9] for recent studies). Still, EAs are often met with scepticism from the RL community. The main argument is that a general purpose optimization technique such as an EA, even if slightly tailored to the learning problem, is not likely to compete with highly specialized methods developed solely for canonical RL scenarios. Strong empirical evidence for the power of evolutionary RL and convincing arguments why certain EAs are particularly well suited for certain RL problem classes are needed to dispel this concern, and this study is a further step in this direction.

Unfortunately, it is not easy to conduct a fair comparison of evolutionary and standard RL techniques. Of course, they can be applied to the same benchmark problems, as for example done in [8,9]. However, usually the search spaces are different, which can introduce a strong bias in the comparison. Many RL algorithms are value function approaches, which learn a function that predicts the expected future reward given a state or an action in a particular state. The policy is then defined on top of this value function [1,2,3]. In contrast, EAs are typically used for direct policy search, that is, they directly optimize a mapping between states and actions. Thus, it is insightful to compare EAs with other methods searching directly in policy space such as policy gradient methods (PGMs), which are well established in the RL community.

We propose variable metric evolution strategies (ESs) for RL [10,11,12,13]. Evolution strategies are per se powerful direct search methods [14]. They usually outperform gradient-based approaches in the presence of noise and on multi-modal objective functions (especially if the local optima have only small basins of attraction). We argue that this makes them particularly well-suited for RL. In RL, noise arises from several sources. The state-transitions and the reward signals may be stochastic. Further, the state observations may be noisy. In addition, the initial state usually varies. This makes it necessary to approximate the quality of a behavioral policy based on a finite number of episodes (or roll-outs). That is, the quality of a policy is a random variable. Evolution strategies adapt the policy as well as parameters of their search strategy (such as the variable metric) based on ranking policies, which is much less error prone than estimating absolute performances or performance gradients [13]. But also for deterministic tasks, evolutionary RL can be advantageous. As we will illustrate in this paper for a non-noisy task, benchmarks problems typically used in RL can be multimodal and are therefore difficult for purely gradient-based methods.

In order to demonstrate the performance of ESs for RL, we compare the covariance matrix adaptation ES (CMA-ES, [15,16]) with random search and a PGM, where we try to make the comparison as fair as possible. Here we consider the natural actor-critic (NAC, [17,18,19]) algorithm, which is an established, state-of-the-art method and our favorite PGM. The NAC is a powerful algorithm for fine-tuning policies and it is arguably one of the best developed and most elaborated PGMs. It is well-suited for comparison with the CMA-ES, because the two algorithms have some conceptual similarities as discussed in [12,13]. In [12,13] NAC and CMA-ES were compared on simple RL problems where the policies had only very few parameters. It is an open question how these results scale with problem dimensionality and difficulty. In this study, we therefore consider convergence speed and success rate on a more difficult variant of the pole balancing problem and take a look at the fitness landscape near optimal solutions. In contrast to EAs, PGMs need a differentiable structure on the space of candidate policies. Here, we consider simple linear policies, which are often used in combination with the NAC.

In the next section, the basic formalism of RL is introduced before we briefly describe the NAC, random weight guessing, and our approach of using the CMA-ES for RL. After that, we describe our experiments in Section 3. Then the results are presented and discussed.

2 Algorithms for Adapting Policy Parameters

Markov decision processes (MDP) are the basic formalism to describe RL problems. An MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ consists of the set of states $\mathcal{S}$, the possible actions $\mathcal{A}$, and for all $a \in \mathcal{A}$ and $s, s' \in \mathcal{S}$ the probabilities $\mathcal{P}^a_{s,s'}$ that action a taken in state s leads to state s' and the expected rewards $\mathcal{R}^a_{s,s'}$ received when going from state s to s' after performing action a. We consider agents interacting with the environment on a discrete time scale. The agent follows its actions according to a behavioral policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where $\pi(s, a)$ is the probability to choose action a in state s (for deterministic policies we write $\pi : \mathcal{S} \to \mathcal{A}$). The goal of RL is to find a policy π such that some notion of expected future reward $\rho(\pi)$ is maximized. For example, for episodic tasks we can define $\rho(\pi) =$

$\sum_{s,s'\in\mathcal{S},a\in\mathcal{A}} d^{\pi}(s)\pi(s,a)\mathcal{P}^{a}_{s,s'}\mathcal{R}^{a}_{s,s'}$, where $d^{\pi}(s)=\sum_{t=0}^{\infty}\gamma^t \Pr\{s_t=s\,|\,s_0,\pi\}$ is the stationary state distribution, which we assume to exist, and s_t is the state in time step t and $\gamma\in]0,1]$ a discount parameter.

In the following, we briefly describe the RL algorithms compared in this study.

2.1 Natural Policy Gradient Ascent

In this section, we introduce the NAC algorithm according to [19]. Policy gradient methods operate on a predefined class of stochastic policies. They require a differentiable structure to ensure the existence of the gradient of the performance measure and ascent this gradient. Let the performance $\rho(\pi)$ of the current policy with parameters $\boldsymbol{\theta}$ be defined as above. Because in general neither d^{π}, $\mathcal{R}$, nor $\mathcal{P}$ are known, the performance gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho(\pi)$ with respect to the policy parameters $\boldsymbol{\theta}$ is estimated from interaction with the environment.

The policy gradient theorem [20] ensures that an unbiased estimate of the performance gradient can be determined from unbiased estimates of the state-action value function $Q^{\pi}(s,a)=\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}|\pi,s_0=s,a_0=a\right]$ (where $r_{t+1}\in\mathbb{R}$ is the reward received after the action in time step t) and the stationary distribution. For any MDP it holds

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho(\pi)=\sum_{s\in\mathcal{S}}d^{\pi}(s)\sum_{a\in\mathcal{A}}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\pi(s,a)Q^{\pi}(s,a). \tag{1}$$

This formulation contains explicitly the unknown value function, which has to be estimated. It can be replaced by a function approximator $f_{\boldsymbol{v}}:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ (the *critic*) with real-valued parameter vector $\boldsymbol{v}$ satisfying the *convergence condition* $\sum_{s\in\mathcal{S}}d^{\pi}(s)\sum_{a\in\mathcal{A}}\pi(s,a)\left[Q^{\pi}(s,a)-f_{\boldsymbol{v}}(s,a)\right]\boldsymbol{\nabla}_{\boldsymbol{v}}f_{\boldsymbol{v}}(s,a)=0$. This leads directly to the extension of the policy gradient theorem for function approximation. If $f_{\boldsymbol{v}}$ satisfies the convergence condition and is *compatible* with the policy parametrization in the sense that

$$f_{\boldsymbol{v}}=\boldsymbol{\nabla}_{\boldsymbol{\theta}}\ln(\pi(s,a))\boldsymbol{v}+\text{const}, \tag{2}$$

then the policy gradient theorem holds if $Q^{\pi}(s,a)$ in (1) is replaced by $f_{\boldsymbol{v}}(s,a)$ [20].

Stochastic policies π with parameters $\boldsymbol{\theta}$ are parametrized probability distributions. The Fisher information matrix $F(\boldsymbol{\theta})$ induces a metric in the space of probability distributions that is independent of the coordinate system [21]. The direction of steepest ascent in this metric space is given by $\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}}\rho(\pi)=F(\boldsymbol{\theta})^{-1}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho(\pi)$, thus inducing "natural" gradient ascent in this direction. We have $F(\boldsymbol{\theta})=\sum_{s\in\mathcal{S}}d^{\pi}(s)\sum_{a\in\mathcal{A}}\pi(s,a)\boldsymbol{\nabla}_{\boldsymbol{\theta}}\ln(\pi(s,a))(\boldsymbol{\nabla}_{\boldsymbol{\theta}}\ln(\pi(s,a)))^{\mathrm{T}}$ using the definitions above. This implies $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho(\pi)=F(\boldsymbol{\theta})\boldsymbol{v}$, which leads to the simple equality $\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}}\rho(\pi)=\boldsymbol{v}$.

The function approximator $f_{\boldsymbol{v}}$ estimates the advantage function $A^{\pi}(s,a)=Q^{\pi}(s,a)-V^{\pi}(s)$, where $V^{\pi}(s)=\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}|\pi,s_0=s\right]$ is the state value function. Inserting this in the Bellman equation for Q^{π} leads to

$$Q^{\pi}(s_t,a_t)=A^{\pi}(s_t,a_t)+V^{\pi}(s_t)=\sum_{s'}P^{a_t}_{s_t,s'}\left(\mathcal{R}^{a_t}_{s_t,s'}+\gamma V^{\pi}(s')\right).$$

Algorithm 1. episodic Natural Actor-Critic

```
1  initialize $\boldsymbol{\theta}$, $\boldsymbol{\Phi} = \mathbf{0}$, $\boldsymbol{R} = \mathbf{0}$, dimension $n$
2  for $k = 1, \ldots$ do
       // $k$ counts number of policy updates
3      for $e = 1, \ldots, e_{max}$ do
           // $e$ counts number of episodes per policy update, $e_{\max} \geq n+1$
4          for $t = 1, \ldots, T$ do
               // $t$ counts number of time steps per episode
5              begin
6                  observe state $\boldsymbol{s}_t$
7                  choose action $a_t$ from $\boldsymbol{\pi_\theta}$
8                  perform action $a_t$
9                  observe reward $r_{t+1}$
10             end
11             for $i = 1, \ldots, n$ do
12                 $\boldsymbol{\Phi}(e,i) \leftarrow \boldsymbol{\Phi}(e,i) + \gamma^t \frac{\partial}{\partial \theta_i} \ln \boldsymbol{\pi_\theta}(s_t, a_t)$
13             $\boldsymbol{R}(e) \leftarrow \boldsymbol{R}(e) + \gamma^t r_{t+1}$
14         $\boldsymbol{\Phi}(e, n+1) \leftarrow 1$
       // update policy parameters:
15     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{R}$
```

Now we sum over a sample path:

$$\sum_{t=0}^{T} \gamma^t A^\pi(s_t, a_t) = \sum_{t=0}^{T} \gamma^t r_{t+1} + \gamma^{T+1} V^\pi(s_{T+1}) - V(s_0).$$

For an episodic task that is in its terminal state in time step T it holds that $V^\pi(s_{T+1}) = 0$, thus, after replacing A^π using (2), we get:

$$\sum_{t=0}^{T} \gamma^t (\boldsymbol{\nabla_\theta} \ln \pi(s_t, a_t))^{\mathrm{T}} \boldsymbol{v} - V(s_0) = \sum_{t=0}^{T} \gamma^t r_{t+1}.$$

For fixed start states we have $V^\pi(s_0) = \rho(\pi)$, and we get a linear regression problem with $n+1$ unknown variables $\boldsymbol{w} = [\boldsymbol{v}^{\mathrm{T}}, V^\pi(s_0)]^{\mathrm{T}}$ that can be solved after $n+1$ observed episodes (where n is the dimension of $\boldsymbol{\theta}$ and $\boldsymbol{v}$):

$$\left[\sum_{t=0}^{T(e_i)} \left(\gamma^t \boldsymbol{\nabla_\theta} \ln \pi(s_t^{e_i}, a_t^{e_i}) \right]^{\mathrm{T}}, -1 \right]^{\mathrm{T}} \boldsymbol{v} = \sum_{t=0}^{T(e_i)} \gamma^t r_{t+1}^{e_i} \;\;, \;\; i = 1, \ldots, n$$

The superscripts indicate the episodes. In Algorithm 1 the likelihood information for a sufficient number of episodes is collected in a matrix $\boldsymbol{\Phi}$ and the return for each episode in $\boldsymbol{R}$. In every update step one inversion of the matrix $\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$ is necessary [19].

2.2 Random Weight Guessing

We use simple random search (random weight guessing) as a baseline comparison [9]. In every iteration new policy parameters are drawn uniformly from an interval $[-\theta_{\max}, \theta_{\max}]^k$, where k is the number of policy parameters. This candidate solution is evaluated and is maintained if it outperforms the best solution so far and discarded otherwise.

2.3 Evolution Strategies

We promote using the CMA-ES for solving MDPs. The highly efficient use of information and the fast adaptation of step size and covariance matrix (which corresponds to learning the metric underlying the optimization problem) makes the CMA-ES one of the best direct search algorithms for real-valued optimization [14]. For a detailed description of the CMA-ES we refer to the articles by Hansen et al. [15,16].

For the first time the CMA-ES was proposed for RL in [10]. It was found that the CMA-ES outperforms alternative evolutionary RL approaches on variants of the pole balancing benchmark in fully and partially observable environments. In a more recent study by [9], these results were compared to 8–12 (depending on the task) other RL algorithms including value-function and policy gradient approaches. On the four test problems where the CMA-ES was considered, it ranked first, second (twice), and third. In [11] the CMA-ES was applied to learn the behavior of a driver assistance system, where neural attractor dynamics were used to represent the policies. In [22] and [23] the CMA-ES was used for RL in robotics. The authors combined the CMA-ES with evolutionary topology optimization to evolve artificial neutral networks.

Recently, we performed a systematic comparison between the CMA-ES and policy gradient methods with variable metrics [12,13]. The preliminary experiments indicate that the CMA-ES is much more robust regarding the choice of hyperparameters and initial policies.

3 Experiments

The experiments conducted in this paper extend our previous work described in [12], where we analyzed the cart pole balancing task, which is a well-known benchmark in RL. In this paper we study a more difficult variant, double-pole balancing, which has already been solved successfully with evolutionary methods, see [24,10,9].

Double-pole balancing. Two poles of different length ($l_1 = 1$m for the first pole with mass $m_1 = 0.1$kg and $l_2 = 0.1$m for the second pole with mass $m_2 = 0.01$kg) are mounted side by side on the same 1-dimensional cart with mass $m_\mathrm{c} = 1$kg and are to be balanced simultaneously. The equations of motion for two poles

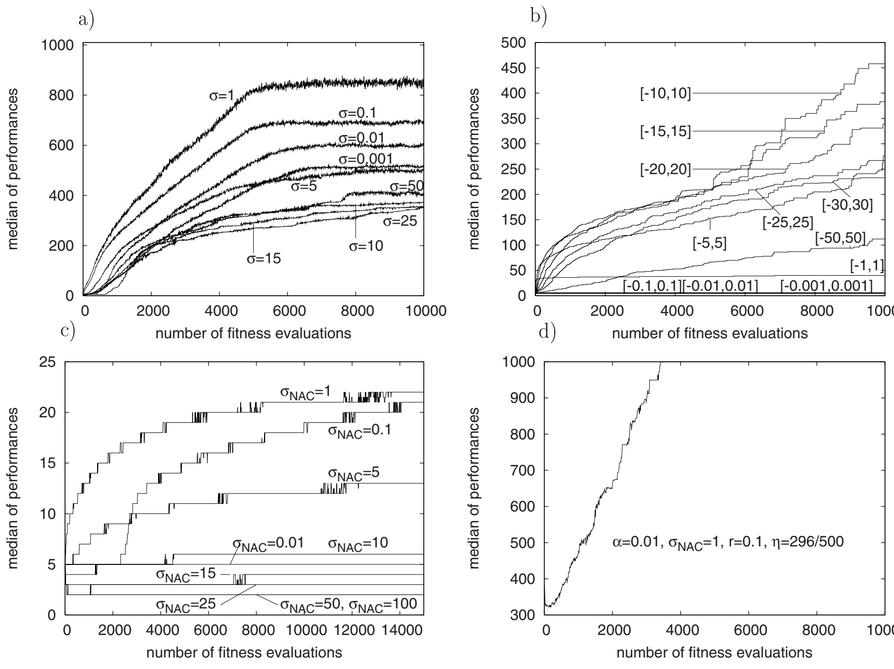


Fig. 1. Performance of CMA-ES, NAC, and random weight guessing on the double pole balancing task. The median over 500 independent trials is shown for the CMA in a) and for random weight guessing in b). For the NAC only exemplary the performance for learning rate $\alpha = 0.1$ is shown in c). The performance of the NAC with initial policy parameters drawn uniformly from a hypersphere with radius $r = 0.1$ centered at a global optimum at $(-4.19408, -13.2605, -1.54318, -36.91, 4.39037, 3.53484)$ and parameter values $\alpha = 0.01$ and $\sigma_{\text{NAC}} = 1$ is shown in d).

are given in [25].[1] This task is only solvable if the two poles differ in length. The state $\boldsymbol{s} = [x, \dot{x}, \zeta_1, \dot{\zeta}_1, \zeta_2, \dot{\zeta}_2]^{\text{T}}$ is given by the cart's distance to the center of the track $x \in [-2.4, 2.4]$ and velocity $\dot{x}$, the current angle ζ_1 of the longer pole and its angular velocity $\dot{\zeta}_1$ and the current angle ζ_2 of the second pole together

[1] For $i \in \{1, 2\}$ we have:

$$\ddot{x} = \frac{F - \mu_{\text{c}}\ \text{sgn}(\dot{x}) + \sum_{i=1}^{2} \tilde{F}_i}{m_{\text{c}} + \sum_{i=1}^{2} \tilde{m}_i}\ ,\quad \ddot{\zeta}_i = -\frac{3}{8\,l_i}\left(\ddot{x}\cos\zeta_i + g\sin\zeta_i + \frac{2\mu_i\,\dot{\zeta}_i}{m_i\,l_i}\right)$$

$$\tilde{F}_i = 2m_i\,l_i\,\dot{\zeta}_i^2\,\sin\zeta_i + \frac{3}{4}\,m_i\,\cos\zeta_i\left(\frac{2\mu_i\,\dot{\zeta}_i}{m_i\,l_i} + g\sin\zeta_i\right)\ ,\quad \tilde{m}_i = m_i\left(1 - \frac{3}{4}\cos^2\,\zeta_i\right)$$

Here $g = 9.81\,\text{m/sec}^2$ is the acceleration due to gravity and $\mu_{\text{c}} = 5 \cdot 10^{-4}\,\text{Ns/m}$ the coefficient of friction of the cart, $\mu_1 = \mu_2 = 2 \cdot 10^{-6}\,\text{Nms}$ are the coefficients of friction for the first and second pole, respectively. The effective force from pole i on the cart is given by $\tilde{F}_i$ and its effective mass by $\tilde{m}_i$. The sign function sgn "inherits" the unit of measurement of its argument.

with its angular velocity $\dot{\zeta}_2$. Actions are continuous forces $a = F$ applied to the cart parallel to the x-axis. The dynamical system is numerically solved using fourth-order Runge-Kutta integration with step size $\tau = 0.01\,\mathrm{s}$.

Experimental setup. The agent follows a deterministic policy $\pi_{\mathrm{deter}}(\boldsymbol{s}) = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{\theta}$, with $\boldsymbol{\theta} \in \mathbb{R}^6$. The policy parameters $\boldsymbol{\theta}$ are initialized with zero.[2] For learning, the NAC uses the stochastic policy $\pi_{\boldsymbol{\theta}}^{\mathrm{stoch}}(\boldsymbol{s}, a) = \mathrm{N}(\pi_{\theta}^{\mathrm{deter}}(\boldsymbol{s}), \sigma_{\mathrm{NAC}})$, where the standard deviation σ_{NAC} is viewed as an additional adaptive seventh parameter of the method and is initialized independently. After every time step the agent receives a reward signal of $r_{t+1} = 1$. A time step corresponds to 0.02s simulation time. An episode ends after 1000 time steps (20s) or when either of the poles leaves the feasible region $[-36°, 36°]$ or the cart leaves the interval $[-2.4, 2.4]$. All episodes start in the same initial state $\boldsymbol{s}_0 = [0, 0, 1°, 0, 0, 0]^{\mathrm{T}}$. Since the task is episodic we use a discount factor of $\gamma = 1$ in the performance measure. The fitness function used by the CMA-ES is the accumulated reward observed over one episode (one episode is sufficient because the task is deterministic in our experiments) $\rho(\pi) = \sum_{t=1}^{T} r_t = T$. Thus the fitness of a policy is determined by the number of time steps T the poles are balanced without the cart leaving the feasible region. The same function is used for evaluation of the NAC and of random weight guessing.

We employ the CMA-ES with rank-μ covariance update [16], where all parameters are set to default values. The population sizes are $\mu = 3$ and $\lambda = 6$, accordingly. Candidate solutions outside the box $[-50, 50]^6$ are discarded and a new offspring is generated. We test different initial global step sizes $\sigma \in \{0.001, 0.1, 1, 1, 10, 15, 20, 25, 50\}$.

In the case of random weight guessing, we vary the interval lengths $\theta_{\max} \in \{0.001, 0.1, 1, 1, 10, 15, 20, 25, 30, 35, 50\}$. For the NAC, we test all combinations of $\alpha \in \{0.0001, 0.001, 0.01, 0.1, 0.3\}$ and $\sigma_{\mathrm{NAC}} \in \{0.1, 1, 5, 10, 15, 25, 50, 100\}$.

Each algorithm gets a budget of 10000 episodes per trial. A trial is stopped and regarded as successful when the poles are balanced for 1000 time steps.

4 Results

Selected results are shown in Fig. 1. Table 1 lists the success rates for the different hyperparameters. The episodic NAC never managed to balance the poles, except for $(\alpha = 0.001, \sigma_{NAC} = 100)$, and $(\alpha = 0.01, \sigma_{NAC} = 50)$, where it found a solution in 1 and 2 out of 500 trials, respectively.

While the double-pole balancing benchmark is not very difficult for evolutionary RL with neural network policies [10,9], it is obviously a challenging problem using linear policies. Although the CMA-ES is usually much better than random weight guessing [13], this difference is not so clear in this study. The performance of both methods depends on the choice of the hyperparameter, the initial global step size and the bounds of the search interval, respectively.

[2] We assume that the measurement units of the single components of $\boldsymbol{\theta}$ are chosen such that $\boldsymbol{s}^{\mathrm{T}}\boldsymbol{\theta}$ is a force.

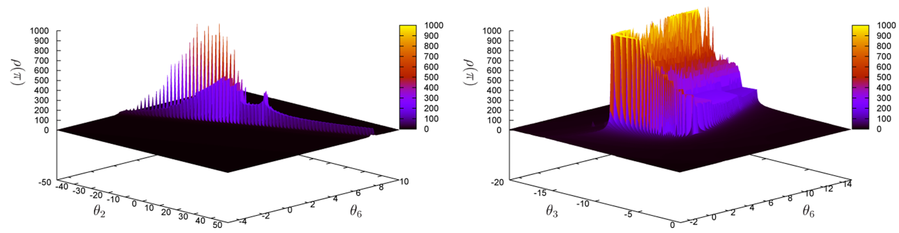


Fig. 2. 2-dimensional projection of the fitness landscape around a global optimum. On the left, the parameters θ_2 and θ_6 are varied while the other parameters are fixed ($\theta_1 = -4.19408$, $\theta_3 = -1.54318$, $\theta_4 = -36.91$, $\theta_5 = 4.39037$). On the right, θ_2 and θ_5 are varied ($\theta_1 = -4.19408$, $\theta_3 = -1.54318$, $\theta_4 = -36.91$, $\theta_6 = 3.53484$).

For $\theta_{\max} \in \{5, 10, 15\}$ random weight guessing succeeds significantly more frequently (χ^2-test, $p < .05$) to balance the poles compared to the CMA-ES with the *worst* hyperparameter choice $\sigma = 20$. For all other values of $\theta_{\max}$ (except $\theta_{\max} \in \{5, 20\}$) the purely random search performed significantly worse (χ^2-test, $p < .05$) than the CMA-ES regardless of the choice of intial global step size σ. The CMA-ES is always significantly better than the episodic NAC. The performance of the CMA-ES is comparatively independent of the choice of the initial step size. Random weight guessing requires that the search intervals fit the problem. If the boundaries are too large, it will not sample a good solution, if they are too small, there might be no good solutions in the parameter ranges at all.

The bad performance of the episodic NAC is striking. To understand the results, we visualized the objective function landscape around a global optimum found by the CMA-ES, see Fig. 2. In these projections, the objective function is clearly multi-modal and contains plateaus of equal (bad) quality. Thus, in these two-dimensional plots the landscapes are almost worst case scenarios for purely gradient-based methods. The objective function is also difficult for ESs, but at least there is some structure in the fitness landscape they can exploit and they are much less likely to get stuck in local optima with small basins of attraction. However, when initializing the policy parameters close to the difficult global optimum shown in Fig. 2 the NAC works very efficiently, see Fig. 1 d).

Table 1. Success rates for CMA-ES and random weight guessing

Success rate η of CMA-ES for different values of the initial global step size σ.										
σ	0.001	0.01	0.1	1	5	10	15	20	25	50
η	$\frac{151}{500}$	$\frac{172}{500}$	$\frac{201}{500}$	$\frac{245}{500}$	$\frac{154}{500}$	$\frac{144}{500}$	$\frac{145}{500}$	$\frac{141}{500}$	$\frac{161}{500}$	$\frac{161}{500}$

Success rate η of stochastic search for different intervals $[-\theta_{\max}, \theta_{\max}]$.												
$\theta_{\max}$	0.001	0.01	0.1	1	5	10	15	20	25	30	35	50
η	$\frac{0}{500}$	$\frac{0}{500}$	$\frac{0}{500}$	$\frac{0}{500}$	$\frac{171}{500}$	$\frac{182}{500}$	$\frac{191}{500}$	$\frac{159}{500}$	$\frac{113}{500}$	$\frac{85}{500}$	$\frac{77}{500}$	$\frac{20}{500}$

5 Conclusion

Evolutionary reinforcement learning (RL) using the covariance matrix adaptation evolution strategy (CMA-ES) resembles policy gradient methods, in particular the episodic natural actor-critic (NAC) algorithm. Both strategies search directly in the space of policies, are variable metric methods, and rely on normally distributed variations for exploration. Of course, the frequency and the level at which the variations are applied vary. We claim that in practice the CMA-ES is much more robust w.r.t. the choice of hyperparameters, policy initialization, and especially noise, while, given appropriate hyperparameters, the NAC can outperform the CMA-ES in terms of learning speed if initialized close to a desired policy. This is supported by the experiments on the double-pole balancing benchmark in this study, which turns out to be surprisingly difficult when linear policies are considered. Because of plateaus and undesired local optima in the objective function landscape, the CMA-ES is superior compared to approaches purely based on estimated performance gradients. However, even the CMA-ES has difficulties on this landscape as shown by the comparison with random search.

In future work we will extend the experiments to different, higher dimensional benchmark tasks and to other direct policy search methods.

Acknowledgment. The authors acknowledge support from the German Federal Ministry of Education and Research within the Bernstein group "The grounding of higher brain function in dynamic neural fields".

References

1. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
2. Bertsekas, D., Tsitsiklis, J.: Neuro-Dynamic Programming. Athena Scientific (1996)
3. Heidrich-Meisner, V., Lauer, M., Igel, C., Riedmiller, M.: Reinforcement learning in a Nutshell. In: 15th European Symposium on Artificial Neural Networks (ESANN 2007), pp. 277–288. d-side publications, Evere (2007)
4. Whitley, D., Dominic, S., Das, R., Anderson, C.W.: Genetic reinforcement learning for neurocontrol problems. Machine Learning 13(2-3), 259–284 (1993)
5. Moriarty, D., Schultz, A., Grefenstette, J.: Evolutionary Algorithms for Reinforcement Learning. Journal of Artificial Intelligence Research 11, 199–229 (1999)
6. Chellapilla, K., Fogel, D.: Evolution, neural networks, games, and intelligence. IEEE Proc. 87(9), 1471–1496 (1999)
7. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
8. Whiteson, S., Stone, P.: Evolutionary function approximation for reinforcement learning. Journal of Machine Learning Research 7, 877–917 (2006)
9. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Efficient non-linear control through neuroevolution. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 654–662. Springer, Heidelberg (2006)

10. Igel, C.: Neuroevolution for reinforcement learning using evolution strategies. In: Congress on Evolutionary Computation (CEC 2003), vol. 4, pp. 2588–2595. IEEE Press, Los Alamitos (2003)
11. Pellecchia, A., Igel, C., Edelbrunner, J., Schöner, G.: Making driver modeling attractive. IEEE Intelligent Systems 20(2), 8–12 (2005)
12. Heidrich-Meisner, V., Igel, C.: Similarities and differences between policy gradient methods and evolution strategies. In: 16th European Symposium on Artificial Neural Networks (ESANN), pp. 149–154. d-side publications, Evere (2008)
13. Heidrich-Meisner, V., Igel, C.: Variable metric reinforcement learning methods applied to the noisy mountain car problem. In: European Workshop on Reinforcement Learning (accepted, 2008)
14. Beyer, H.G.: Evolution strategies. Scholarpedia 2(8), 1965 (2007)
15. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
16. Hansen, N., Müller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
17. Peters, J., Vijayakumar, S., Schaal, S.: Reinforcement learning for humanoid robotics. In: Proc. 3rd IEEE-RAS Int'l Conf. on Humanoid Robots, pp. 29–30 (2003)
18. Riedmiller, M., Peters, J., Schaal, S.: Evaluation of policy gradient methods and variants on the cart-pole benchmark. In: Proc. 2007 IEEE Internatinal Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007), pp. 254–261 (2007)
19. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing 71(7-9), 1180–1190 (2008)
20. Sutton, R., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, vol. 12, pp. 1057–1063 (2000)
21. Amari, S., Nagaoka, H.: Methods of Information Geometry. Translations of Mathematical Monographs, vol. 191. American Mathematical Society and Oxford University Press (2000)
22. Siebel, N.T., Sommer, G.: Evolutionary reinforcement learning of artificial neural networks. International Journal of Hybrid Intelligent Systems 4(3), 171–183 (2007)
23. Kassahun, Y., Sommer, G.: Efficient reinforcement learning through evolutionary acquisition of neural topologies. In: 13th European Symposium on Artificial Neural Networks, d-side, pp. 259–266 (2005)
24. Gomez, F., Miikkulainen, R.: Solving non-Markovian control tasks with neuroevolution. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, pp. 1356–1361 (1999)
25. Wieland, A.: Evolving neural network controllers for unstable systems. In: IJCNN 1991-Seattle International Joint Conference on Neural Networks, 1991, vol. 2 (1991)

Optimal Nesting of Species for Exact Cover: Many against Many

Jeffrey Horn

Northern Michigan University, Marquette MI 49855, USA
jhorn@nmu.edu
http://cs.nmu.edu/~jeffhorn

Abstract. Experiments with resource-defined fitness sharing (RFS) applied to shape nesting problems indicate a remarkable ability to discover exact covers of resources [1, 2]. These exact covers are represented by a maximally sized set of cooperating (non-competing) species. Recent papers by Horn [3, 4] introduce the first formal analyses of this empirical phenomenon. In [3], a minimal case of two species, **a** and **b**, against a third, **c**, is considered: the *two-against-one* scenario. It is shown that if the team of **a** and **b** form an exact cover, then **c** will be extinct at niching equilibrium. In [4], this result is generalized to the case of *two-against-many*: if **a** and **b** form an exact cover against an arbitrary number of competing species, under very general assumptions, **a** and **b** will be the only survivors at niching equilibirum. In the current paper, we extend these results to the most general scenario: *many-against-many*. We prove that, under certain very general assumptions, any size team of species forming an exact cover will dominate a population with any number of competing species: at niching equilibirum, all such competitors will be extinct. The results are more general than shape-nesting problems, applying as well to the NP-complete problem *exact cover by k-sets*.

1 Introduction

This paper completes a series of three analytical papers that attempt to explain the unusual results reported in the 2002 paper [1] that introduced resource-defined fitness sharing (RFS)[1]. In the 2002 paper, the RFS niching method exhibits a robust ability to converge to an optimal solution on shape nesting problems if that optimal solution consists of a *tiling* (i.e., *exact cover*). Since RFS operates with quantities defined by sets (e.g., set intersections), this ability to tile shapes generalizes to an ability to find exact covers of arbitrary sets. That is, if an exact cover can be found in the current population, it appears that RFS will always drive the population distribution to represent the exact cover.

Two previous papers begin a formal analysis of this empirical phenomenon. The first [3] examines the minimal case of "cooperation versus competition", namely the *two-against-one* case. In this scenario two species cooperate to compete against one other species. The two species form an exact cover of all resources, while the third species is completely overlapped (in resource coverage)

[1] RFS, applied to shape nesting problems, is the subject of U.S. Patent No. 7,181,702.

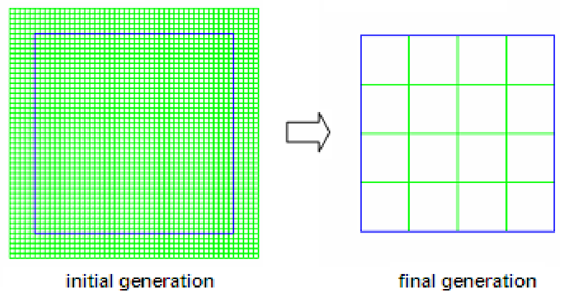


Fig. 1. RFS can select for *tilings* of a surface

by the combination of the first two species. The second paper [4] generalizes this analysis to the case of *two-against-many*, in which two cooperating species compete against an arbitrary number of other species. Again, the two cooperative species cover all of the resources, while the others compete for coverage. In this paper we finish generalizing this analysis by investigating the *many-against-many* case, in which any number of species form an exact cover and compete against arbitrarily many other species.

2 Background

We summarize the RFS algorithm and the problem domain of shape nesting, which is a subset of resource covering problems in general. This summary is meant to motivate the analysis at the heart of this paper but is not essential to understanding the analysis. Horn [1] gives details about the origin of RFS.

2.1 RFS Applied to Shape Nesting

Shape nesting algorithms attempt to place shaped pieces on a finite substrate so as to maximize the number of such pieces on the substrate with no overlaps [5, 6]. On one particular test problem, Horn [1] applies RFS to a two-dimensional shape nesting problem limited to axis-aligned squares. The width of the substrate square is four times that of the piece square, so that a single optimal solution exists, consisting of sixteen pieces exactly covering the substrate, as shown in Figure 1, right. Starting with a random population of 16,000 square pieces (Figure 1, left), the GA with RFS is able to select and promote the sixteen *species* corresponding to the solution in Figure 1, right, where each of the 16 species is represented by approximately 1000 copies (individuals) in the final population.

2.2 The RFS Algorithm

The fitness of each individual in the current population is calculated as follows. Each chromosome specifies a placement of a piece. If that placement causes a

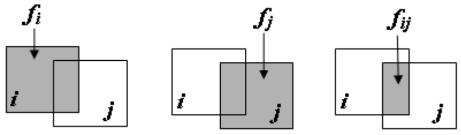


Fig. 2. The basic terms used in specifying RFS

piece to extend beyond the boundaries of the substrate, the individual is assigned a fitness of 0. All other individuals (i.e., chromosomes specifying piece placements entirely on the substrate), receive a shared fitness greater than 0, for use in a standard selection method (e.g., tournament selection, proportionate selection).

Individual Fitness. The shared fitness for each individual depends on the amount of resources (e.g., area) covered by the individual, and on the amount that coverage overlaps with that of other individuals in the population. The RFS shared fitness formula, $f_{sh,i}$, takes the form of a fraction:

$$f_{sh,i} = \frac{f_i}{niche\_count(i)} = \frac{f_i}{\sum_{j \in P} f_{ij}}, \tag{1}$$

where i is an individual in population P, f_i is the objective (unshared) fitness of i, and f_{ij} is the pairwise overlap in "coverage" between individuals i and j in P, and $niche\_count(i)$ is a measure of the degree of competition for resources covered by i. Under RFS, niche count is defined as the cumulative pairwise overlap between i and other individuals in P. Figure 2 illustrates the terms f_i, f_j, and f_{ij} for two individuals i and j.

Species Fitness. Next we define what we mean by *species* as opposed to individuals. We consider a species to be a set of identical individuals (this means identical coverage of resources). Thus each unique chromosomes defines a unique species. Any two members of the same species overlap completely, while between any two members of different species there is less than complete overlap .

We can now re-write Equation 1 in terms of species:

$$f_{sh,x} = \frac{f_x}{niche\_count(x)} = \frac{f_x}{\sum_{y \in S(P)} n_y f_{xy}}. \tag{2}$$

Equations 2 and 1 are equivalent. In Equation 1, the summation in the niche count is taken over the population of individuals (using the variable j). In Equation 2, the population is partitioned into a set $S(P)$ of species y, with $y \in S(P)$. The shared fitness for any member of a species x is thus equal to the objective fitness of that species divided by the niche count for that species. The species niche count is equal to the sum over all species of the interaction term (f_{xy}) multiplied (weighted) by the number of members of that species (i.e., the *species count*: n_y) in the current population P.

Identical Coverage. This paper considers only the situations in which individuals (and therefore species) have identical objective fitnesses (e.g., the nesting of identical shapes, or the selection of fixed size subsets for set covering). Therefore we normalize the objective fitness f_i to 1, $\forall i \in P$. Thus $0 \leq f_{ij} \leq 1, \forall i, j \in P$, and $f_{sh,i} = \frac{1}{niche\_count(i)}$.

2.3 A Static Analysis of Niching Equilibrium

A population distribution is said to be at evolutionary equilibrium if application of the selection operator yields the same distribution (in expectation) [7]: $E[p_x(t+1)] = p_x(t)$, where $p_x(t)$ is the proportion of species x in the population at time t, and $E[p_x(t+1)]$ is the expected proportion of x at time $t+1$ (e.g., the next generation). Under RFS with proportionate selection, this equation implies that at equilibrium the shared fitness $f_{sh,x}$ of all species must be equal to the average fitness, and therefore must be equal to each other: $\forall(x, y \in S(P)) : f_{sh,x} = f_{sh,y}$. Since we assume that the objective fitnesses are identical, then equilibrium requires that the niche counts must all be equal: $\forall_{x \in S(P)} \sum_{y \in S(P)} n_y f_{xy} = C'$, where C' is some constant (actually the inverse of the average fitness at equilibrium).

We assume the infinite population model to allow the existence of an exact solution to the niching equilibrium equations, and to avoid an integer programming problem in our analysis. Thus our equilibrium equations become $\forall_{x \in S(P)} \sum_{y \in S(P)} p_y f_{xy} = C'$, where $0 \leq p_y \leq 1$ is the proportion of species y in current population P. Finally, we have an additional equation that helps determine a unique solution, namely, $\sum_{x \in S(P)} p_x = 1$.

Before proceeding with the static analysis, we note that to determine the true stability of an equilibrium state, and to find all such states, a dynamic analysis is needed. In particular, Friedman [8] uses dynamic analysis in the context of evolutionary games to illustrate different types of dynamic and static equilibria, and to relate them to one another. Dynamic analysis of niching equilibrium is beyond the scope of this paper but is an important direction in which to extend its results.

Three Species: Two Against One. In [3] the entire population P is divided up among three species: $S(P) = \{a, b, c\}$. If p_a, p_b, and p_c are the proportions of the population for species a, b, and c respectively, then $p_a + p_b + p_c = 1$, and at equilibrium,

$$\begin{aligned}
niche\_count(a) &= p_a f_{aa} + p_b f_{ab} + p_c f_{ac} = p_a + p_b f_{ab} + p_c f_{ac} = C' \\
niche\_count(b) &= p_a f_{ab} + p_b f_{bb} + p_c f_{bc} = p_a f_{ab} + p_b + p_c f_{bc} = C' \\
niche\_count(c) &= p_a f_{ac} + p_b f_{bc} + p_c f_{cc} = p_a f_{ac} + p_b f_{bc} + p_c = C'
\end{aligned}$$

Before attempting to solve these equations, Horn [3] further constrains the overlap (a.k.a., interaction) terms f_{xy}: he focuses on the case of "exact cover" by two out of three species (e.g., Figure 3, top left). Two properties follow from the exact coverage by a and b:

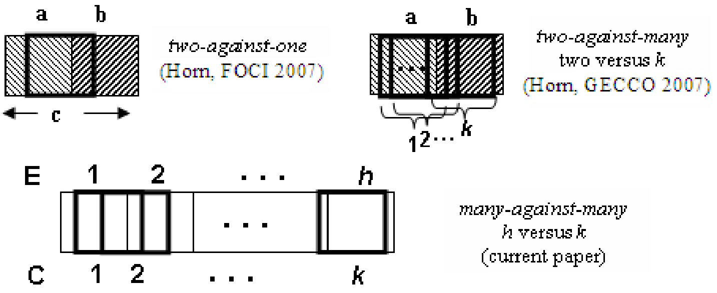


Fig. 3. Situations in which Properties I and II hold

Property I: *Minimum* $a \leftrightarrow b$ *Competition*	$f_{ab} = 0$
Property II: *Maximum* $(a, b) \leftrightarrow c$ *Competition*	$f_{ac} + f_{bc} = 1$

Under Property II species **c** is completely *covered* by **a** and **b**.

These two properties allow Horn to solve uniquely the set of niching equilibrium equations to find that $(p_a, p_b, p_c) = (\frac{1}{2}, \frac{1}{2}, 0)$. Thus if an exact cover of resources exists in a population, then under RFS selection only the species representing the exact cover can be expected to resist invasion by another species at niching equilibrium (with the other species extinct at niching equilibrium).

Many Species: Two Against k. Horn [4] generalizes his previous results (revisited above) to niching scenarios in which the exact cover team of species a and b compete against an arbitrary number, $k > 0$, of competing (i.e., overlapping) species. This scenario is depicted in Figure 3, top right.

To handle an arbitrary number of competitors, Horn drops the use of letters for the competing species and instead numbers the k competing species $1..k$. Horn [4] also introduces matrices to represent the set of equations defining niching equilibrium. He proceeds to solve the set of equations to show that at equilibrium, the competing species are extinct while the two exact cover species, a and b, split the population.

Because the generalization in [4] to k competitors is so similar to our new generalization to h exact cover species versus k competitors, we simply present our new results and derivation without repeating the details in [4].

3 Generalization to Many EC Species: h against k

We now try to generalize the results of [4] to niching scenarios in which the exact cover set consists of an arbitrary number $h > 0$ of species. Once again, these exact cover (EC) species compete against an arbitrary number, $k > 0$, of competing (i.e., overlapping) species. We extend our notation thus: the set of species S in the population P is partitioned into two subsets, $E \bigcup C = S(P)$, where E is the set of species comprising the exact cover of resources and C is the set of species that compete with E by overlap. Let $\|E\| = h$ and $\|C\| = k$. Figure 3, bottom, shows an example scenario.

3.1 Interaction Matrices

We re-write the species interaction matrix $\mathbf{M}_{RFS}$ introduced in [4], and consider a helpful partition:

$$\mathbf{M}_{RFS} = \left[\begin{array}{cccc|cccc} f_{E_1E_1} & f_{E_1E_2} & \cdots & f_{E_1E_h} & f_{E_1C_1} & f_{E_1C_2} & \cdots & f_{E_1C_k} \\ f_{E_2E_1} & f_{E_2E_2} & \cdots & f_{E_2E_h} & f_{E_2C_1} & f_{E_2C_2} & \cdots & f_{E_2C_k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{E_hE_1} & f_{E_hE_2} & \cdots & f_{E_hE_h} & f_{E_hC_1} & f_{E_hC_2} & \cdots & f_{E_hC_k} \\ \hline f_{E_1C_1} & f_{E_2C_1} & \cdots & f_{E_hC_1} & f_{C_1C_1} & f_{C_1C_2} & \cdots & f_{C_1C_k} \\ f_{E_1C_2} & f_{E_2C_2} & \cdots & f_{E_hC_2}, & f_{C_1C_2} & f_{C_2C_2} & \cdots & f_{C_2C_k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{E_1C_k} & f_{E_2C_k} & \cdots & f_{E_hC_k} & f_{C_1C_k} & f_{C_2C_k}, & \cdots & f_{C_kC_k} \end{array}\right] = \left[\begin{array}{c|c} EE & EC \\ \hline CE & CC \end{array}\right]. \quad (3)$$

The upper left submatrix $\mathbf{M}_{EE}$ consists of all interactions solely among the exact-cover species E. The lower right submatrix $\mathbf{M}_{CC}$ contains all interactions solely between pairs of competing species C. The lower left and upper right submatrices are transposes of each other, and define the interactions between each exact cover species and each competitor species.

We note that this square matrix is symmetric about the main diagonal, since niche overlap is a symmetric relationship: $f_{xy} = f_{yx}$. We therefore choose to always write the species subscripts in the same order, with exact cover species listed before competitor species (e.g., $f_{E_3C_2}$ is written rather than $f_{C_2E_3}$). We further note that the entries on the main diagonal are all 1, since $\forall X \in S(P) : f_{xx} = 1$. Generalizing Property I (minimal competition among EC species) from [4], gives $f_{E_iE_j} = f_{E_jE_i} = 0$, for $i \neq j$. Thus our matrix $\mathbf{M}_{RFS}$ above can be re-written as the left-hand side of Equation 4 below.

The submatrix $\mathbf{M}_{EE}$ in the $\mathbf{M}_{RFS}$ of Equation 4 is an identity matrix, thanks to Property I. Furthermore, we can generalize Property II (maximum E versus C competition), to $\forall_{i\in(1..k)} \sum_{j=1}^{h} f_{E_jC_i} = 1$. This means that the sum of each column in the upper right submatrix $\mathbf{M}_{EC}$, and the sum of each row in the lower left submatrix $\mathbf{M}_{CE}$, are equal to one.

The lower right submatrix $\mathbf{M}_{CC}$ contains all interactions solely among the competing species $(C_1, ..., C_k)$. There are no apparent implications of Properties I or II for this submatrix, because the interactions there are independent of the species in set E.

3.2 Niching Equilibrium Matrix

Under RFS, the matrix $\mathbf{M}_{RFS}$ essentially "defines" the niche counts. Recall that at niching equilibrium, the niche counts (and hence the shared fitness) of all species present in the population must be equal:

$$\left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & f_{E_1C_1} & f_{E_1C_2} & \cdots & f_{E_1C_k} \\ 0 & 1 & \dots & 0 & f_{E_2C_1} & f_{E_2C_2} & \cdots & f_{E_2C_k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & f_{E_hC_1} & f_{E_hC_2} & \cdots & f_{E_hC_k} \\ \hline f_{E_1C_1} & f_{E_2C_1} & \cdots & f_{E_hC_1} & 1 & f_{C_1C_2} & \cdots & f_{C_1C_k} \\ f_{E_1C_2} & f_{E_2C_2} & \cdots & f_{E_hC_2} & f_{C_1C_2} & 1 & \cdots & f_{C_2C_k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{E_1C_k} & f_{E_2C_k} & \cdots & f_{E_hC_k} & f_{C_1C_k}, & f_{C_2C_k} & \cdots & 1 \end{array}\right] \left[\begin{array}{c} p_{E_1} \\ p_{E_2} \\ \vdots \\ p_{E_h} \\ \hline p_{C_1} \\ p_{C_2} \\ \vdots \\ p_{C_k} \end{array}\right] = \left[\begin{array}{c} C' \\ C' \\ \vdots \\ C' \\ \hline C' \\ C' \\ \vdots \\ C' \end{array}\right]. \tag{4}$$

where C' is some constant. That is, at niching equilibrium, the niche count for every species is the same ($\forall x \in S, niche\_count_{eq}(x) = C'$).

3.3 Solving the Niching Equilibrium Equations

We proceed to solve the above system of linear equations for the general case of k competitors, using Gauss-Jordan elimination.

We note that the zeroing of the first column in the $\mathbf{M}_{CE}$ submatrix is accomplished by subtracting $f_{E_1C_{row}}$ times the first row in $\mathbf{M}_{RFS}$ from each "competitor row" *row* in $\mathbf{M}_{RFS}$ (that is, $1 \leq row \leq k$). We can continue the Gauss-Jordan elimination procedure in this manner, zeroing out the column *col* in the CE submatrix by subtracting $f_{E_{col}C_{row}}$ times the col^{th} row in $\mathbf{M}_{RFS}$ from each competitor row *row* in $\mathbf{M}_{RFS}$ (that is, $1 \leq row \leq k$ and $1 \leq col \leq h$).

After zeroing all h columns in the $\mathbf{M}_{CE}$ (lower left) submatrix in this manner, we are left with

$$\left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & f_{E_1C_1} & f_{E_1C_2} & \cdots & f_{E_1C_k} \\ 0 & 1 & \dots & 0 & f_{E_2C_1} & f_{E_2C_2} & \cdots & f_{E_2C_k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & f_{E_hC_1} & f_{E_hC_2} & \cdots & f_{E_hC_k} \\ \hline 0 & 0 & \dots & 0 & 1-\sum_{i=1}^{h} f_{E_iC_1}f_{E_iC_1} & f_{C_1C_2}-\sum_{i=1}^{h} f_{E_iC_1}f_{E_iC_2} & \cdots & f_{C_1C_k}-\sum_{i=1}^{h} f_{E_iC_1}f_{E_iC_k} \\ 0 & 0 & \dots & 0 & f_{C_1C_2}-\sum_{i=1}^{h} f_{E_iC_2}f_{E_iC_1} & 1-\sum_{i=1}^{h} f_{E_iC_2}f_{E_iC_2} & \cdots & f_{C_2C_k}-\sum_{i=1}^{h} f_{E_iC_2}f_{E_iC_k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & f_{C_1C_k}-\sum_{i=1}^{h} f_{E_iC_k}f_{E_i,C_1} & f_{C_2C_k}-\sum_{i=1}^{h} f_{E_iC_k}f_{E_iC_2} & \cdots & 1-\sum_{i=1}^{h} f_{E_iC_k}f_{E_iC_k} \end{array}\right] \tag{5}$$

More succinctly, we can say that after enough Gauss-Jordan elimination steps, the submatrices $\mathbf{M}_{EE}$ and $\mathbf{M}_{EC}$ remain unchanged, while submatrix $\mathbf{M}_{CE}$ consists of all zeros, and the submatrix $\mathbf{M}_{CC}$ has entries

$$f_{C_{row}C_{col}} - \sum_{i=1}^{h} f_{E_iC_{row}} f_{E_iC_{col}}$$

for the (row, col) entry of $\mathbf{M}_{CC}$. On the right-hand side, the column vector looks like this:

$$\left[\begin{array}{c} C' \\ C' \\ \vdots \\ C' \\ \hline C' - \sum_{i=1}^{h} C' * f_{E_iC_1} \\ C' - \sum_{i=1}^{h} C' * f_{E_iC_2} \\ \vdots \\ C' - \sum_{i=1}^{h} C' * f_{E_iC_k} \end{array}\right], \tag{6}$$

where the right-hand side entry for each row *row* (for the last k rows) is

$$C' - \sum_{i=1}^{h} C' * f_{E_iC_{row}} \Rightarrow C' - C' * \sum_{i=1}^{h} f_{E_iC_{row}}.$$

By Property II, $\sum_{i=1}^{h} f_{E_iC_{row}} = 1$ for $row = 1..k$, simplifying our expression above to zero: $C' - C' * 1 = 0$. Placing these right-hand side zeros into our niching equilibrium matrix equation, submatrix $\mathbf{M}_{CC}$ yields

$$\begin{bmatrix} 1 - \sum_{i=1}^{h} f_{E_iC_1} f_{E_iC_1} & f_{C_1C_2} - \sum_{i=1}^{h} f_{E_iC_1} f_{E_iC_2} & \cdots & f_{C_1C_k} - \sum_{i=1}^{h} f_{E_iC_1} f_{E_iC_k} \\ f_{C_1C_2} - \sum_{i=1}^{h} f_{E_iC_2} f_{E_iC_1} & 1 - \sum_{i=1}^{h} f_{E_iC_2} f_{E_iC_2} & \cdots & f_{C_2C_k} - \sum_{i=1}^{h} f_{E_iC_2} f_{E_iC_k} \\ \vdots & \vdots & \ddots & \vdots \\ f_{C_1C_k} - \sum_{i=1}^{h} f_{E_iC_k} f_{E_iC_1} & f_{C_2C_k} - \sum_{i=1}^{h} f_{E_iC_k} f_{E_iC_2} & \cdots & 1 - \sum_{i=1}^{h} f_{E_iC_k} f_{E_iC_k} \end{bmatrix} \tag{7}$$
$$* \begin{bmatrix} p_{C_1} \\ p_{C_2} \\ \vdots \\ p_{C_k} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

We can see that submatrix $\mathbf{M}_{CC}$ above represents a homogenous system of linear equations, since the values on the right-hand side are all zero. If the original matrix $\mathbf{M}_{RFS}$ is non-singular, then so is this submatrix, and there is a unique solution to it. This solution must be the trivial solution:

$$p_{C_1} = p_{C_2} = ... = p_{C_k} = 0.$$

Substituting this solution into the E rows of our matrix above (that is, the first h rows), we find that $p_{E_1} = p_{E_2} = ... = p_{E_h} = C'$. If we now make the *single, infinite population assumption*: $\sum_{\forall x \in S(P)} p_x = 1$. For us this means that

$$p_{E_1} = p_{E_2} = ... = p_{E_h} = C' = \frac{1}{h}.$$

Thus we have shown that under the assumption that the niching interaction matrix is non-singular, if an exact cover exists in the population, then the only solution to the niching equilibrium equations has the equilibrium population uniformly distributed among the species in the exact cover set E. All other species (those not in the set E) are extinct at niching equilibrium.

Theorem 1. *If a set E of distinct species exactly cover resources, then under RFS with proportionate selection and an infinite population, a sufficient condition for E to take over the population at niching equilibrium, is the non-singularity of the niching interaction matrix.*

PROOF: The proof is given above.

4 Discussion

One major result of our analysis is the conclusion that if a set E of species together exactly cover the resources of any and all other species, and if these species form the only exact cover, then these species will take over the population at niching equilibrium, resisting "invasion" by any and all other "redundant" species. This is remarkable, in that $k + h$ species can all have the same *objective* fitness; they each cover the same amount of resources. Yet RFS selection strongly favors the h against the other k. This preference must be due solely to the greater coverage of one particular ensemble of species over any other.

The limitations of this conclusion arise from the assumptions made in the analysis. For example, the use of proportions instead of actual numbers means we are using the "infinite population" model, in which we assume that the population is large enough to exactly realize any proportions generated by the manipulation of the equations in our model. Such an assumption can be tested by experiment and by dynamic analysis, both of which are missing in the current study. A dynamic analysis is of particular importance. Our static analysis does not preclude the existence of dynamic attractors, such as cycles, or of other complex, dynamic behaviors. Another limitation of our model is the assumption that only a single, exact cover exists. What about "ties"? Our model does not preclude them, only our assumptions do. It might not be difficult to extend the analysis to consider problems with multiple solutions, or to modify RFS to deal with multiple solutions. Another limitation with deep implications is the non-singularity of the niching interaction matrix $\mathbf{M}_{RFS}$. Under what conditions is this matrix nonsingular? Does the non-singularity condition translate to meaningful conditions in the physical world of niche overlaps?

We have also not considered exploration operators such as recombination and mutation. While these can be added to the analysis in future work, it may be that these are applied sparingly enough to have neglible effect on the dynamics of RFS and selection. We should consider also the implication of the current analysis that RFS and selection alone can solve a hard search problem (i.e., exact cover) and so could be used without variational operators. That is, with a sufficiently large and diverse initial population, using RFS to select the optimal (covering) subset would by itself be a significant evolutionary computation.

A final observation: the RFS algorithm and the results herein are not limited to spatial "nesting" of geometric shapes. They apply to the nesting of any kind of set. Shape overlap is really just a special case of set intersection. Thus the most general problem domain to which this analysis is applicable is *exact cover by k-sets*[2].

References

[1] Horn, J.: Resource-based fitness sharing. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN VII 2002. LNCS, vol. 2439, pp. 381–390. Springer, Heidelberg (2002)

[2] This is a known NP-Complete problem [9]. k is the fixed size of the subsets, which implements our assumption of identical shapes/areas in RFS.

[2] Horn, J.: Coevolving species for shape nesting. In: Schaffer, J.D. (ed.) The 2005 IEEE Congress on Evolutionary Computation (IEEE CEC 2005), Piscataway, NJ, pp. 1800–1807. IEEE Computer Society Press, Los Alamitos (2005)
[3] Horn, J.: Optimal nesting of species for exact cover of resources: Two against one. In: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), pp. 322–330. Omnipress (2007)
[4] Horn, J.: Optimal nesting of species for exact cover of resources: Two against many. In: Proceedings of the 2007 Genetic And Evolutionary Computation Conference (GECCO 2007), pp. 448–455. The Association For Computing Machinery (2007)
[5] Kendall, G.: Applying Meta-Heuristic Algorithms to the Nesting Problem Utilising the No Fit Polygon. PhD thesis, University of Nottingham (2000)
[6] Dighe, R., Jakiela, M.J.: Solving pattern nesting problems with genetic algorithms: employing task decomposition and contact detection between adjacent pieces. Evolutionary Computation 3(3), 239–266 (1996)
[7] Maynard-Smith, J.: Evolution and the Theory of Games. Cambridge University Press, Cambridge (1982)
[8] Friedman, D.: Evolutionary games in economics. Econometrica 59(3), 637–666 (1991)
[9] Garey, M.R., Johnson, D.S.: Computers and Intractability: a Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)

Nonsynonymous to Synonymous Substitution Ratio k_a/k_s: Measurement for Rate of Evolution in Evolutionary Computation

Ting Hu and Wolfgang Banzhaf

Department of Computer Science, Memorial University of Newfoundland, Canada
{tingh,banzhaf}@cs.mun.ca

Abstract. Measuring fitness progression using numeric quantification in an Evolutionary Computation (EC) system may not be sufficient to capture the rate of evolution precisely. In this paper, we define the rate of evolution R_e in an EC system based on the rate of efficient genetic variations being accepted by the EC population. This definition is motivated by the measurement of "amino acid to synonymous substitution ratio" k_a/k_s in biology, which has been widely accepted to measure the rate of gene sequence evolution. Experimental applications to investigate the effects of four major configuration parameters on our rate of evolution measurement show that R_e well reflects how evolution proceeds underneath fitness development and provides some insights into the effectiveness of EC parameters in evolution acceleration.

1 Introduction

Evolutionary computation is a method that simulates natural evolution to search for solutions to optimization problems. This field has seen significant progress in the past decades. Improving the evolutionary capabilities of an evolutionary system has attracted substantial attention recently [1], particularly enabling an evolutionary computation system to generate *evolvable* adaptation to environments. Measuring the rate of evolution can help to quantify evolutionary capabilities, and thus can be used to accelerate evolution through designing better computation models. At the time of writing, the rate of evolution has not yet seen a formal definition in the literature other than measuring fitness progression over generations. At first glance, a definition reflecting how fast an evolutionary population is improving its fitness may seem sufficient. However, considered as the capability to generate adaptation, evolutionary progress cannot be determined by how good population fitness is per se, but should be regarded as a "second-order" effect of fitness improvements. Therefore, we believe that the rate of evolution should be better defined by looking beyond fitness and should be measured by the rate of genetic variations being generated and accepted.

Some methods to quantify evolutionary capabilities have been proposed in the literature. Bedau and Packard [3] proposed a method to identify the capabilities of creating adaptation during evolutionary processes. It is based on calculating *evolutionary activity* statistics of components in an evolutionary system. Their comparison between artificial and natural evolutionary systems by studying evolutionary activities has shown that the "long-term" trend of generating adaptation is deficient in artificial systems, i.e., the capability of generating evolvable adaptation is not as strong in artificial evolutionary systems.

In molecular evolution biologists use the k_a/k_s ratio to measure the evolution rate of gene sequences [8,9]. Such a measurement compares two homologous protein-coding gene sequences from two related species. The k_a/k_s ratio resulting from measuring the number of nonsynonymous (amino acid) substitutions per nonsynonymous site (k_a) to the number of synonymous substitutions per synonymous site (k_s) characterizes the rate of evolution between these two sequences. Here, *substitutions* only include those observable genetic changes having been accepted into the gene sequences. Since k_s measures neutral evolution (without involving functional improvements under selection pressure), the k_a/k_s ratio reflects the rate of *adaptive* evolution against the *background* rate of evolution. This measurement has been widely applied in the analysis of adaptive molecular evolution, and is regarded as a general method of measuring the rate of sequence evolution in biology.

In this paper, we transfer the measurement of this k_a/k_s ratio to EC. We utilize a Genetic Programming system to implement measuring the rate of evolution. Specifically, the rate of evolution in a GP system and the measurement of this rate are defined here. Comparative experiments on varying parameters, including tournament selection size, population size, mutation rate, and crossover rate, show the effectiveness of this approach. It is able to capture the rate of generating adaptive variations, which cannot be well observed in fitness development. We conclude this paper with a brief discussion on some future research.

2 The k_a/k_s Ratio in Biology

In molecular biology, a *codon* in DNA consists of three nucleotides, and each codon determines one amino acid of a corresponding protein. A sequence of amino acids forms a protein, which produces the functional phenotype of an organism. A single nucleotide substitution on a codon produces another codon. Due to the redundancy of the genetic code, different codons may encode the same amino acid (e.g., codons *AAA* and *AAG* both code for amino acid lysine). Thus, a nucleotide substitution on a codon may be *synonymous*and not produce an amino acid replacement. A different nucleotide substitution may produce different amino acids, and this codon change is a *nonsynonymous* (amino acid) change. To characterize each site on a codon, in particular, for a codon ε, if $f_\varepsilon(i)$ $(i = 1, 2, 3)$ denotes the fraction of nonsynonymous single-nucleotide substitutions among all possible single-nucleotide substitutions at site i, therefore, the number of nonsynonymous sites on codon ε is $\sum_{i=1}^{3} f_\varepsilon(i)$, and subsequently, the number of synonymous sites on codon ε is $3 - \sum_{i=1}^{3} f_\varepsilon(i)$ [8].

Biologists compare two homologous protein-coding nucleotide gene sequences from related species. These two relevant sequences carry similar genes, i.e., are homologous. However, there can be differences at some nucleotide loci as a result of evolution. Some of these differences on the two gene sequences may result in generating different amino acids for encoding proteins, i.e., are nonsynonymous substitutions, and some of them may not modify the proteins, i.e., are synonymous. The differences between two homologous gene sequences are counted by pairwise comparison of codons. Specifically, the number of nonsynonymous nucleotide substitutions is denoted by M_{a}, and that of synonymous nucleotide substitutions is M_{s}. Further, the total number of nonsynonymous (synonymous, resp.) sites for an entire gene sequence is calculated by summing up all the numbers of nonsynonymous (synonymous, resp.) sites on each codon. For the two comparative gene sequences, N_{a} means the average number of nonsynonymous sites of two sequences. Similarly, N_{s} is obtained as the number of synonymous sites. Therefore, the nonsynonymous substitution rate $k_{\mathrm{a}} = M_{\mathrm{a}}/N_{\mathrm{a}}$ is the number of observed nonsynonymous substitutions divided by the total number of such type of changes that these sequences *are capable of.* This is a metric of how much evolution has occurred in protein sequences normalized by all possible genetic variations between the two species. Rate $k_{\mathrm{s}} = M_{\mathrm{s}}/N_{\mathrm{s}}$ is the number of observed synonymous changes divided by the total number of such changes that the sequences are capable of. This metric measures the "background" rate of "silent" genetic evolution without phenotypical improvement between the two species.

Therefore, the ratio $k_{\mathrm{a}}/k_{\mathrm{s}}$ quantifies the rate of evolution by stating efficient evolutionary changes in relation to silent background evolutionary changes. This ratio also reflects the selection pressure on the evolution of organisms. In the case of $k_{\mathrm{a}}/k_{\mathrm{s}} > 1$, fixation of nonsynonymous substitutions is faster than that of synonymous substitutions, which means that *positive selection* fixes amino acid changes faster than silent ones. While mostly one finds $k_{\mathrm{a}}/k_{\mathrm{s}} < 1$, the case where deleterious substitutions are eliminated by *purifying selection* (negative selection), and the rate of fixation of amino acid changes is reduced. If $k_{\mathrm{a}} = k_{\mathrm{s}}$, the fixation of these two types of changes are at the same rate. Measuring a large $k_{\mathrm{a}}/k_{\mathrm{s}}$ ratio suggests that adaptive genetic variations have been generated and fixed at a high rate.

3 Measuring Rate of Evolution in EC

Inspired by the $k_{\mathrm{a}}/k_{\mathrm{s}}$ measurement on the rate of sequence evolution in biology, we define the rate of evolution and propose a measurement method for EC systems. An EC system better capable of evolution can generate efficient adaptation under selection pressure, so it has a *potential* to improve fitness. Evidently, this capability or potential is less observable than fitness itself. Since fast evolution is caused by generating adaptations at a high rate we can focus on the adaptive genetic changes underneath the phenotypical fitness to investigate the evolutionary progress. Here, we define the *rate of evolution* R_{e} as the rate of adaptive genetic changes being accepted into an EC system. Since selection acts at the phenotypical level, the adaptation of a genetic change to its environment can be determined by its acceptance into the population. Some changes that are able to improve the adaptation will be accepted, i.e., nonsynonymous substitutions, while other attempted deleterious changes will be eliminated. Some silent changes will be accepted as synonymous substitutions without experiencing selection pressure on phenotypical improvement. Dividing the rate of adaptive substitutions by the rate of synonymous substitutions can quantify the rate of adaptive evolution in an EC system. Therefore, if selection favors the innovated adaptive genetic changes at a high rate relative to the background rate, we say that this EC system has a high rate of evolution.

As a case study, we utilize a tree-based GP system to implement this idea because GP individuals possess similar features to gene sequences. For example, for a GP tree in our case, genetic changes can be nonsynonymous as in biological systems, which lead to representing different functions, or synonymous, which keep the encoded function unchanged. We calculate the number of substitutions and divide it by the "sites" for a GP system to obtain the two types of rates. Here, we measure the rate of evolution for a GP system in each generation. Specifically, before establishing a generation t, standard mutation and crossover, limited to subtree replacement, are applied to the individual trees in a GP population of generation $t - 1$. Truncation tournament selection is then performed on both the parents and offspring to form the next generation t. In such an iteration, we define the rate of evolution $R_\mathrm{e}(t)$ of generation t by observing the individual genetic changes and their acceptance into the population.

It is well known that changes to a GP tree may be silent due to the existence of neutral *intron* codes [2]. That is, syntactic changes to a tree may or may not lead to functional changes. Therefore, after mutation or crossover of the trees, these subtree replacements are either nonsynonymous or synonymous. For each individual tree i, if a change is silent, the value of nonsynonymous change $m_\mathrm{a}^i(t)$ is set to 0 and the value of synonymous change $m_\mathrm{s}^i(t)$ is set to 1. In contrast, if a change leads to functional differences, $m_\mathrm{a}^i(t)$ is 1 and $m_\mathrm{s}^i(t)$ is 0. If tree i is not modified from generation $t-1$ to generation t, both $m_\mathrm{a}^i(t)$ and $m_\mathrm{s}^i(t)$ remain 0. After the truncation tournament selection chooses new individuals from both the parents and offspring, a new generation t is established. As a result, the total number of nonsynonymous substitutions $M_\mathrm{a}(t)$ and synonymous substitutions $M_\mathrm{s}(t)$ for the entire population of generation t can be calculated as

$$M_\mathrm{a}(t) = \sum_{i=1}^{S} m_\mathrm{a}^i(t) \ , \quad M_\mathrm{s}(t) = \sum_{i=1}^{S} m_\mathrm{s}^i(t) \ , \tag{1}$$

where S is the population size. Note that, $M_\mathrm{a}(t)$ and $M_\mathrm{s}(t)$ only count those genetic changes accepted into the population, i.e., substitutions, which have survived through the selection.

As we discussed in the biological $k_\mathrm{a}/k_\mathrm{s}$ ratio definition (Sect. 2), the numbers of nonsynonymous sites and synonymous sites represent the *potential* of the sequence to produce nonsynonymous or synonymous changes, and are used to "normalize" the numbers of substitutions. Here, we adopt a *sensitivity* notion to describe the potential of a GP tree to change its semantic meaning in the event of a subtree replacement. Trees have varying sensitivities against subtree replacements, an observation made by Langdon and Banzhaf [6] in research on repeated patterns in tree-based GP systems. We keep a record of all changes to a tree from the beginning of evolution including all attempted subtree replacements, such that the accumulated fraction of these changes being nonsynonymous or synonymous can be regarded as the nonsynonymous sensitivity and synonymous sensitivity of this tree. Specifically, for an individual tree i after initialization, we use $c_{\mathrm{a}}^i(t)$ and $c_{\mathrm{s}}^i(t)$ to denote the accumulated numbers of nonsynonymous and synonymous changes of generation t, respectively, obtained by summing up all the previously recorded changes that have happened to this tree,

$$c_{\mathrm{a}}^i(t) = c_{\mathrm{a}}^i(t-1) + m_{\mathrm{a}}^i(t) \ , \quad c_{\mathrm{s}}^i(t) = c_{\mathrm{s}}^i(t-1) + m_{\mathrm{s}}^i(t) \ , \tag{2}$$

with

$$c_{\mathrm{a}}^i(0) = c_{\mathrm{s}}^i(0) = 0 \ . \tag{3}$$

Therefore, the nonsynonymous and synonymous sensitivities of tree i of generation t can be obtained as follows from the fraction of each type of changes, and these metrics indicate the degree of tree i being changed nonsynonymously or synonymously,

$$n_{\mathrm{a}}^i(t) = \frac{c_{\mathrm{a}}^i(t)}{c_{\mathrm{a}}^i(t) + c_{\mathrm{s}}^i(t)} \ , \quad n_{\mathrm{s}}^i(t) = \frac{c_{\mathrm{s}}^i(t)}{c_{\mathrm{a}}^i(t) + c_{\mathrm{s}}^i(t)} \ . \tag{4}$$

We add up the sensitivities of all individuals in the population to obtain the total nonsynonymous and synonymous sensitivities as the “sites” of the current generation,

$$N_{\mathrm{a}}(t) = \sum_{i=1}^{S} n_{\mathrm{a}}^i(t) \ , \quad N_{\mathrm{s}}(t) = \sum_{i=1}^{S} n_{\mathrm{s}}^i(t) \ . \tag{5}$$

Last, we define the nonsynonymous and the synonymous substitution rates k_{a} and k_{s} of generation t as

$$k_{\mathrm{a}}(t) = \frac{M_{\mathrm{a}}(t)}{N_{\mathrm{a}}(t)} \ , \quad k_{\mathrm{s}}(t) = \frac{M_{\mathrm{s}}(t)}{N_{\mathrm{s}}(t)} \ . \tag{6}$$

The rate $k_{\mathrm{a}}(t)$ measures the rate of generating nonsynonymous adaptive changes. The rate $k_{\mathrm{s}}(t)$ describes the rate of producing neutral changes in an evolutionary process. Without changes at the functional level, these neutral changes will not experience pressure in evolution. Thus, $k_{\mathrm{s}}(t)$ practically provides “clock ticks” for the acceptance of genetic changes in the GP system. Since $k_{\mathrm{a}}(t)$ measures the rate of accepted effective changes, the ratio $k_{\mathrm{a}}(t)/k_{\mathrm{s}}(t)$ represents the “evolutionary distance” in relation to the “evolutionary time”, therefore, the rate of effective adaptation of generation t. Thus, we propose the rate of evolution R_{e} in the GP tree population of generation t to be

$$R_{\mathrm{e}}(t) = \frac{k_{\mathrm{a}}(t)}{k_{\mathrm{s}}(t)} \ . \tag{7}$$

4 Experimental Results

As a demonstration, we calculate R_e using GP to solve a benchmark quintic polynomial symbolic regression problem $x^5 - 2x^3 + x$ defined by Koza [5]. Each individual in this GP population is a syntax tree initialized by the method *ramped half-and-half* with maximum depth 6. Candidate functions are evolved toward a target function $f(x) = x^5 - 2x^3 + x$ within interval $[-1, 1]$ by matching a set of sample points. The sample set has 50 real numbers uniformly distributed in $[-1, 1]$. The absolute difference between output and the target $f(x)$ value is the error, and the fitness function is defined as the average error over all 50 samples. The terminal set includes variable x and random ephemeral constants generated from 2001 numbers equally distributed in $[-1, 1]$ with granularity of 0.001. The four arithmetic operators: $+$, $-$, $\times$, and protective $\div$ are used as the function set. We apply random mutation and crossover with probabilities 0.1 and 0.9, respectively, and the maximum mutation subtree depth is 4. Parent individuals and offspring after genetic changes compete through truncation selection with tournament size 4. This GP system has a population size of 4000 evolved for a maximum of 50 generations. A set of 20 cases are used as inputs to a GP tree before and after mutation or crossover, to test whether a subtree replacement is nonsynonymous or synonymous. If all 20 cases produce the same output, subtree replacement applied to this tree is regarded synonymous; otherwise, this tree is considered to have undergone a nonsynonymous change.

A preliminary experiment of this rate of evolution measurement on a single GP evolutionary process can be found in Hu and Banzhaf [4]. Here, we compare R_e in different configuration scenarios by varying such parameters as selection size, population size, mutation rate, and crossover rate, to study their effects on the rate of evolution and to verify the effectiveness of our approach. In each set of experiments, we only change the investigated parameter and hold all others constant. The average fitness, k_a, k_s and R_e are plotted with the average values of 50 successful runs. The method *exponentially weighted moving average* is used here to smooth the curves (smoothing factor 0.1).

4.1 Tournament Selection Size

We increase tournament selection size from 4 to 6 and to 8 (Fig. 1). It is generally accepted that a larger tournament selection size generates greater survival pressure, and thus can maintain a better fitness in the population. It can be seen that the population under tournament selection size 8 has the best average fitness. However, due to a higher selection pressure, fewer innovative individuals are accepted, so the population with tournament size 8 has the lowest nonsynonymous substitution rate k_a. In contrast, relatively more silent changes are accepted with a larger tournament selection size. This also concurs with a recent prediction by Luke and Panait [7] that bloat of neutral code in GP is caused by the pressure of improving fitness. Therefore, the rate of evolution R_e decreases as the tournament size increases. These results show that higher selection pressure slows down the rate of accepting genetic variations.

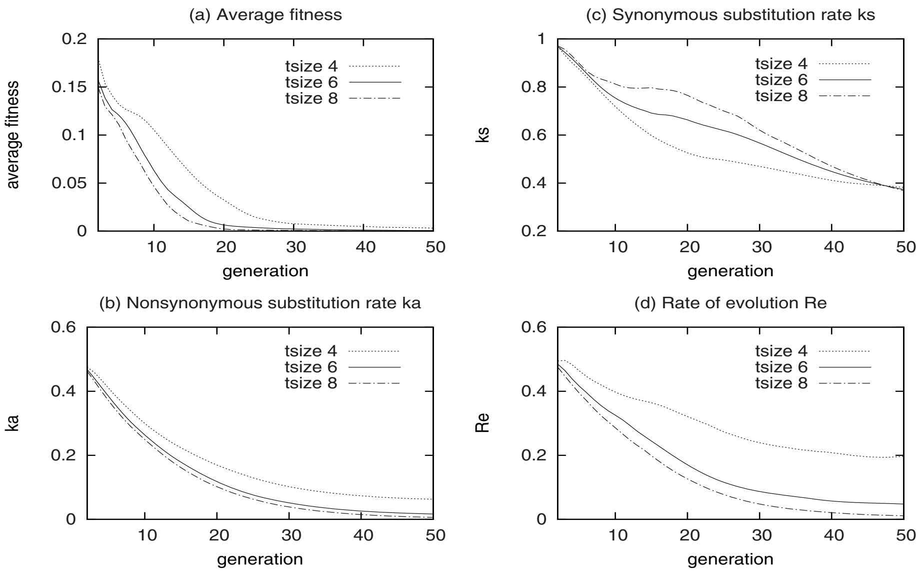


Fig. 1. Rate of evolution with different selection pressure

4.2 Population Size

We test the GP system with different population sizes 200, 2,000 and 20,000 (Fig. 2). Observe that a larger population is better at searching and maintaining the average fitness. All three nonsynonymous substitution rates k_a with different population sizes are quite close, which indicates that, although larger populations offer a larger amount of adaptive individuals to be generated and accepted, their rates in this static symbolic regression problem are nearly the same as smaller populations. Further, a larger population accepts synonymous genetic changes at a slower rate, which is an expected result of a slower propagating speed of dominant individuals. It can be observed that a larger population has a slightly higher R_e at the early stage of the search process but slows down when the target individual becomes dominant in the population. These differences are quite small, however, for this static optimization problem. So we believe that, although a larger population offers more chances of innovating adaptation, under the same environment and selection pressure, a larger population does not have a real advantage in improving the rate of evolution. It can be seen further that the population with size 200 has the most drastically changing rates, accepting genetic changes at a fairly high rate even around generation 50 (see also the average fitness chart).

4.3 Mutation Rate

The mutation rate is set to 0.3, 0.6, and 0.9 when the crossover rate is fixed to 0.1 (Fig. 3). In our simulations, we only collect successful runs which can

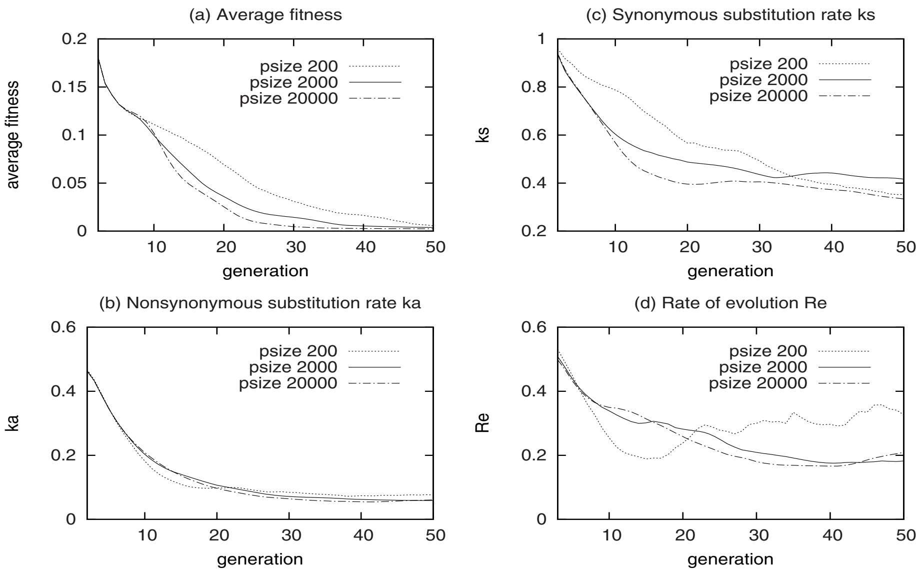


Fig. 2. Rate of evolution with different population sizes

reach the target function within 50 generations. A population with a higher mutation rate is more likely to succeed. We observed that the percentages of successful runs with mutation rates 0.3, 0.6, and 0.9 are 16%, 22%, and 30%. However, despite different success likelihoods, various mutation rates do not show significant differences in the rate of improving the average fitness solving this problem. In our rate of evolution measurement, it can be observed that, a higher mutation rate results in a higher nonsynonymous substitution rate k_a and a lower synonymous rate k_s, and thus, a higher evolution rate R_e. These results show that a higher mutation rate can accelerate evolution but also brings in more noise at the end of evolution (Fig. 3 (d)). Moreover, this simulation supports a general tendency of mutation to maintain high population diversity.

4.4 Crossover Rate

In this set of simulations, we fix the mutation rate at 0.1 and increase the crossover rate from 0.3 to 0.6, and to 0.9. In Fig. 4, similarly to varying mutation rates, we can see that investigating fitness development is not sufficient for drawing conclusions on the effectiveness of crossover rate on the rate of adaptive evolution. In our measurement, it is observed that a larger crossover rate provides more adaptive genetic changes, i.e., a greater k_a, and consequently a higher rate of evolution R_e. However, the differences between mutation and crossover operations are their effects on synonymous substitution rate k_s. That is, increasing the crossover rate can result in a higher synonymous rate, which implies that crossover contributes more to neutral evolution than mutation.

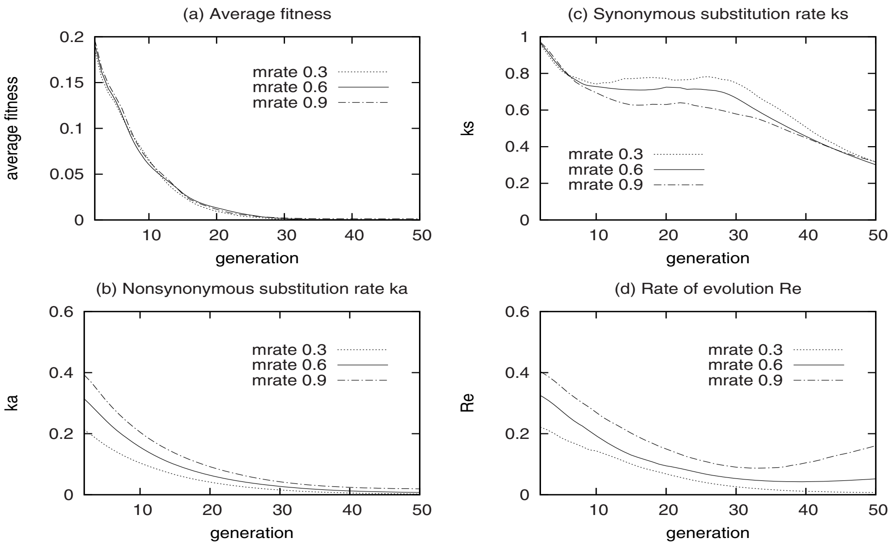


Fig. 3. Rate of evolution with different mutation rates

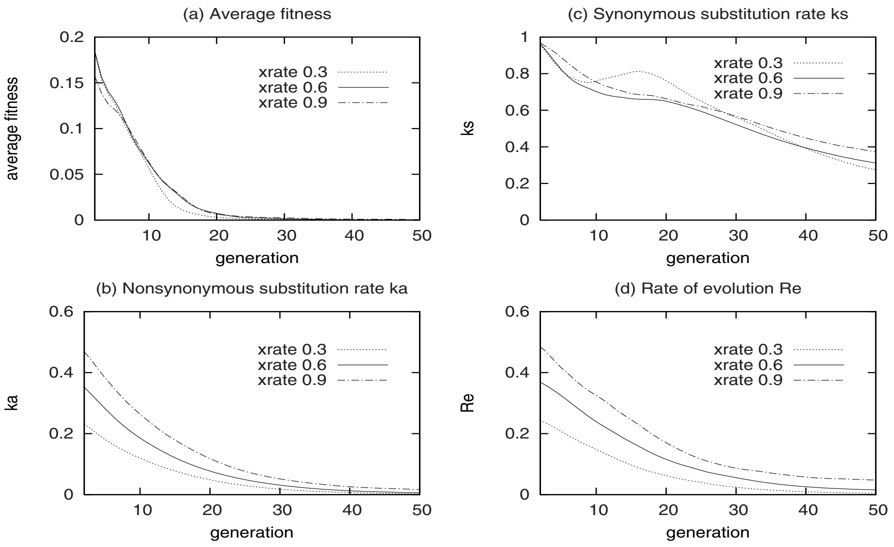


Fig. 4. Rate of evolution with different crossover rates

5 Conclusion and Future Work

In this paper, we introduced the equivalent of a biological measurement of the nonsynonymous to synonymous substitution ratio k_a/k_s. The experimental applications show the ability of this measurement to capture the rate of generating efficient genetic variations in an EC system. Therefore, we believe that the rate of evolution should be better defined by looking beyond fitness and should be measured by the rate of adaptation being generated and accepted. Further, some observations show that in the truncation selection scheme tournament size, mutation rate, and crossover rate are directly related to the rate of evolution, while population size has an indirect relation.

The R_e measurement can be extended in different ways. First, this measurement can be used to help determine adaptive population size in EC. Through visualizing the rate of evolution at different stages, adaptive population size can be chosen to provide effective diversity. Therefore, population size can be chosen systematically rather than empirically. Second, applications of this measurement to various methods in evolutionary computation need to be thoroughly investigated. Third, we propose to use this method for quantification of evolvability since it can reflect the evolutionary capabilities of an artificial evolutionary system.

Acknowledgements

The authors acknowledge support from NSERC Discovery Grant under RGPIN 283304-07.

References

1. Banzhaf, W., Beslon, G., Christensen, S., Foster, J.A., Kepes, F., Lefort, V., Miller, J.F., Radman, M., Ramsden, J.J.: From artificial evolution to computational evolution: A research agenda. Nature Reviews Genetics 7(9), 729–735 (2006)
2. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming: An Introduction On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, San Francisco (1998)
3. Bedau, M.A., Packard, N.H.: Measurement of evolutionary activity, teleology, and life. In: Artificial Life II, pp. 431–461. Addison-Wesley, Redwood City (1992)
4. Hu, T., Banzhaf, W.: Measuring rate of evolution in genetic programming using amino acid to synonymous substitution ratio k_a/k_s. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO 2008), Atlanta, GA, pp. 1337–1338 (2008)
5. Koza, J.R.: Genetic programming II: automatic discovery of reusable programs. MIT Press, Cambridge (1994)
6. Langdon, W.B., Banzhaf, W.: Repeated patterns in tree genetic programming. In: Proceedings of the 8th European Conference on Genetic Programming, Lausanne, Switzerland, pp. 190–202 (2005)
7. Luke, S., Panait, L.: A Comparison of Bloat Control Methods for Genetic Programming. Evolutionary Computation 14(3), 309–334 (2006)
8. Miyata, T., Yasunaga, T.: Molecular evolution of mRNA: A method for estimating evolutionary rates of synonymous and amino acid substitutions from homologous nucleotide sequences and its application. Journal of Molecular Evolution 16(1), 23–36 (1980)
9. Yang, Z., Bielawski, J.P.: Statistical methods for detecting molecular adaptation. Trends in Ecology and Evolution 15(12), 496–503 (2000)

Examining the Effect of Elitism in Cellular Genetic Algorithms Using Two Neighborhood Structures

Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima

Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
hisaoi@cs.osakafu-u.ac.jp, nori@ci.cs.osakafu-u.ac.jp, nojima@cs.osakafu-u.ac.jp
http://www.ie.osakafu-u.ac.jp/~hisaoi/ci_lab_e

Abstract. Elitism has a large effect on the search ability of evolutionary algorithms. Many studies, however, did not discuss its different implementations in cellular algorithms. Usually a replacement policy called "replace-if-better" is applied to each cell in cellular algorithms as a kind of elitism. In this paper, we examine three implementations of elitism. One is global elitism where a pre-specified number of the best individuals in the entire population are viewed as being the elite. The replace-if-better policy is applied only to the globally best individuals. Another scheme is local elitism where an individual is viewed as being the elite if it is the best among its neighbors. The replace-if-better policy is applied only to the locally best individuals. The other scheme is cell-wise elitism where the replace-if-better policy is applied to all individuals. Effects of elitism are examined through computational experiments using a cellular genetic algorithm with two neighborhood structures. One is for local competition among neighbors. This competition neighborhood is used in the local elitism to determine the locally best individuals. The other is for local selection of parents. This selection neighborhood is also called the mating neighborhood. Since we have the two neighborhood structures, we can specify the size of the competition neighborhood for the implementation of the local elitism independent of the selection neighborhood for mating. Experimental results show that the use of the replace-if-better policy at all cells is not always the best choice.

1 Introduction

Elitism usually improves the search ability of evolutionary algorithms. For example, it is well-known that elitism is indispensable in the design of efficient evolutionary multiobjective optimization algorithms [6], [22]. A possible negative effect of elitism is the decrease in the diversity of individuals due to high selection pressure. For example, Jagerskupper and Storch [13] discussed positive aspects of non-elitist algorithms such as the ability to escape from local optima. In this paper, we examine the effect of elitism on the search ability of cellular genetic algorithms using three different implementations of elitism.

Cellular algorithms are one of the most popular models of spatially structured evolutionary algorithms [2], [4]. Since early studies in the late 1980s [8], [14] and the early 1990s [18], [19], cellular algorithms have been an active research area (e.g., see [1]-[4], [7]). In cellular algorithms, each individual is spatially fixed in a cell of a lattice (typically a two-dimensional grid-world). A new offspring in a cell is generated from individuals in its neighboring cells. The main characteristic feature of cellular algorithms is the use of local selection, which is based on a neighborhood structure. It was shown in the literature [9], [15], [16] that the size of the neighborhood structure has a large effect on the behavior of cellular algorithms.

Whereas a single neighborhood structure has been usually used in cellular algorithms in the literature, some studies were based on two neighborhood structures. For example, evolution of altruism in a two-dimensional grid-world was actively studied under the name of structured demes in the late 1970s [5], [17], [20], [21] using two neighborhood structures: One is for local competition and the other is for local selection. Recently two neighborhood structures have been used to analyze the evolution of cooperative behavior in spatial prisoner's dilemma games (e.g., [10], [12]). The use of two neighborhood structures was also examined for function optimization problems in [11] where good results were obtained from the combination of a small competition neighborhood and a large selection neighborhood.

In this paper, we examine different implementations of elitism in a cellular genetic algorithm with two neighborhood structures. A replacement policy called "replace-if-better" has often been used at all cells in cellular algorithms as a kind of elitism in the literature. This policy replaces an individual in a cell with the generated offspring only if the latter has a better fitness value. In this paper, we use the replace-if-better policy together with a random tiebreak mechanism: When the current individual and the offspring have the same fitness value, the replacement is performed with the probability of 0.5. The replace-if-better policy is a kind of elitism at each cell. In this paper, an "elite" individual means an individual to which the replace-if-better policy is applied. On the other hand, a non-elite individual is always replaced with the generated offspring independent of their fitness values. We examine three implementations of elitism: global, local and cell-wise elitism. These three implementations are different in the specification of elite individuals.

The cell-wise elitism is the same as the replacement scheme in standard cellular algorithms where the replace-if-better policy is applied to all cells (i.e., to all individuals). In the global elitism, a prespecified number of the best individuals in the entire population are the elite. The replace-if-better policy is applied only to the globally best individuals. On the other hand, an elite individual is the locally best individual among all neighbors in its competition neighborhood in the local elitism. That is, the replace-if-better policy is applied only to the locally best individuals.

The intensity of the global elitism is specified by the number of elite individuals, which is a user-definable parameter. An extreme case of the global elitism is the same as the cell-wise elitism where the number of elites is the population size. On the other hand, the intensity of the local elitism is specified by the size of the competition neighborhood. The cell-wise elitism can be also viewed as an extreme case of the local elitism with the minimum competition neighborhood of size one. In this case, all individuals are locally best because there is only a single individual in the competition neighborhood. Another extreme case of the local elitism, in which the entire population is used as the competition neighborhood, is a kind of the global elitism.

In this paper, we first explain our cellular genetic algorithm with two neighborhood structures in Section 2. Next we examine its performance through computational experiments in Section 3. It is shown that the choice of an implementation scheme of elitism has a dominant effect on the performance of our cellular genetic algorithm. It is also shown that the choice of an appropriate implementation scheme heavily depends on the problem and the parameter specifications. That is, the use of the replace-if-better policy at all cells is not always the best choice. Finally we conclude this paper in Section 4.

2 Cellular Genetic Algorithms with Two Neighborhood Structures

We use a two-dimensional grid-world where a single individual is spatially fixed in each cell. Thus the number of cells is the same as the population size. We assume the torus structure in our two-dimensional grid-world. In Fig. 1, we show some typical examples of neighborhood structures used in the literature. In each plot, open circles are the neighbors of the closed circle individual.

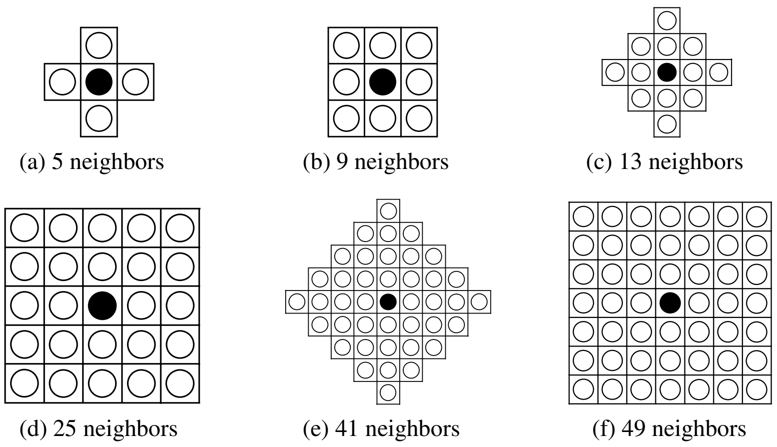


Fig. 1. Six neighborhood structures examined in this paper

As we have already mentioned, we use two neighborhood structures in our cellular genetic algorithm. One is for local competition among neighbors. This neighborhood structure determines the neighbors against which each individual competes. We denote the competition neighborhood of the ith cell as $N_{\text{Compete}}(i)$. The ith cell itself is included in $N_{\text{Compete}}(i)$. The competition neighborhood is used only for the definition of the local elitism in this paper whereas it was also used for recalculating the fitness value of each individual in our former study [11]. The rank of each individual in its competition neighborhood was used as its recalculated fitness value in [11].

The other neighborhood structure is for local selection. This neighborhood structure determines the neighbors from which two parents are chosen to generate an offspring. The selection neighborhood is also called the mating neighborhood. We denote the selection neighborhood of the ith cell as $N_{\text{Select}}(i)$. The ith cell itself is included in $N_{\text{Select}}(i)$. We use binary tournament selection to select two parents from

$N_{\mathrm{Select}}(i)$ for generating an offspring for the ith cell. If the current individual in the ith cell is not an elite individual, it is always replaced with the offspring. On the other hand, the replace-if-better policy is applied to the current individual if it is an elite individual. The replacement of individuals is performed in a synchronized manner.

We use the six neighborhood structures in Fig. 1 for local competition and local selection. All the 6×6 combinations of them are used in computational experiments.

In this paper, we examine three implementations of elitism: global, cell-wise and local elitism. In the following, we explain each implementation.

Global elitism. A prespecified number of the best individuals in the entire population are handled as elite individuals in the global elitism. The global elitism is independent of the two neighborhood structures. The replace-if-better policy is applied only to the globally best individuals in the global elitism.

Cell-wise elitism. The replace-if-better policy is applied to all cells (i.e., all individuals). That is, all individuals are handled as elite individuals in the cell-wise elitism. The cell-wise elitism is also independent of the two neighborhood structures.

Local elitism. The implementation of the local elitism depends on the competition neighborhood. When an individual has the highest fitness value among its neighbors in the competition neighborhood, it is handled as an elite individual. That is, the replace-if-better policy is applied only to the locally best individuals in the local elitism.

We examine three versions of the local elitism, which are different in the handling of the tie situation where the current individual in a cell has the same locally best fitness value as some neighbors in its competition neighborhood. Let K be the number of neighbors (including the current individual in the cell) that have the same locally best fitness value in the competition neighborhood. Version 1 handles the locally best individual as an elite individual only when it is better than all the other neighbors (i.e., when $K = 1$). Version 2 probabilistically determines whether the locally best individual is an elite individual or not in the tie situation. The probability of being an elite individual is specified as the inverse of the number of the locally best neighbors (including the current individual). That is, the probability is specified as $1/K$. Version 3 always handles the locally best individual as an elite individual even when some other neighbors (or all neighbors) have the same locally best fitness value.

3 Performance Evaluation of Cellular Genetic Algorithms

We used a 500-item 0/1 knapsack problem with two constraint conditions. This problem was generated from the original two-objective 500-item knapsack problem in [22] by defining an integrated fitness function as $fitness(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$ where $\mathbf{x}$ is a 500-dimensional binary vector, $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are the two objectives of the original problem in [22]. Each individual (i.e., solution) is represented by a binary string of length 500. Thus the size of the search space is 2^{500}.

Our cellular genetic algorithm was used under the following specifications:

Grid-world: 11×11 (i.e., population size: 121),
Crossover probability (uniform crossover): 0.8,
Mutation probability (bit-flip mutation): 1/(string length),
Stopping condition: 2000 generations.

In the global elitism, we examined four specifications of the number of the global elite individuals: 1 (1% of the population size), 6 (5%), 12 (10%) and 24 (20%). In the same manner as [22], we used the maximum profit/cost ratio-based greedy repair scheme to transform infeasible solutions into feasible ones. This repair scheme was implemented in the Lamarckian manner.

Using the optimal solution **x*** of our test problem, we calculated the relative error of the obtained solution **x** as

$$Relative\ error(\mathbf{x}) = \frac{fitness(\mathbf{x}^*) - fitness(\mathbf{x})}{fitness(\mathbf{x}^*)} \times 100\ (\%). \tag{1}$$

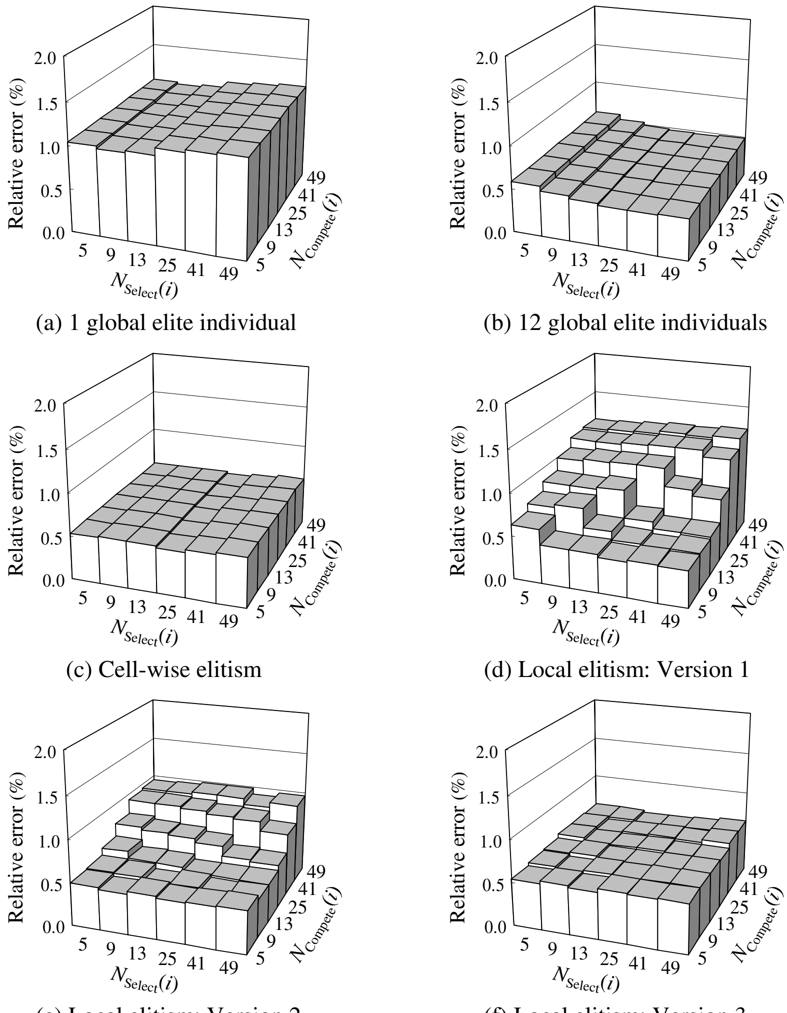
The average relative error was calculated over 100 runs of our cellular genetic algorithm after the 2000th generation for each setting about elitism and neighborhood. Experimental results are summarized in Fig. 2 where the base plane of each plot shows the size of the two neighborhood structures (i.e., the x-axis is the selection neighborhood $N_{\mathrm{Select}}(i)$ while the y-axis is the competition neighborhood $N_{\mathrm{Compete}}(i)$). We can see from Fig. 2 that the choice of an implementation scheme of elitism has a dominant effect on the search ability of our cellular genetic algorithm. For example, good results were not obtained from the global elitism with only a single elite individual in Fig. 2 (a). Since good results were obtained in Fig. 2 (b), we can see that the number of elite individuals in Fig. 2 (a) was too small. When the size of the competition neighborhood was too large in Version 1 and Version 2 of the local elitism, good results were not obtained, either. This is also because the number of elite individuals was too small. In Fig. 3, we show the average percentage of elite individuals over 100 runs with 2000 generations for Version 1 and Version 3 of the local elitism. From the comparison between Fig. 2 and Fig. 3, we can see that good results were not obtained by the local elitism when the percentage of elite individuals was too small.

In Fig. 2, good results were obtained independent of the choice of neighborhood structures when we used the global elitism with 12 elite individuals in (b), the cell-wise elitism in (c), and Version 3 of the local elitism in (f). That is, those implementation schemes of elitism were robust with respect to the size of neighborhood structures. On the contrary, Version 1 and Version 2 of the local elitism were sensitive to the size of neighborhood structures. The best result in Fig. 2 (i.e., 0.410 highlighted in bold print in Table 1), however, was obtained from local elitism Version 1 with a small competition neighborhood and a large selection neighborhood. This observation was consistent with our former study on function optimization problems [11].

In Table 1, we show the average relative error together with the corresponding standard deviation (in the parentheses) obtained from the best specification of the neighborhood structures for each implementation scheme of elitism. It should be noted that the competition neighborhood was used only in the local elitism. (Thus the experimental results in Fig. 2 (a)-(c) were flat with respect to the y-axis: the size of the competition neighborhood).

In Table 1, the best result was obtained from local elitism Version 1. Using Welch's t-test, we compared the best result of local elitism Version 1 (i.e., 0.410) and that of the cell-wise elitism (i.e., 0.524) in Table 1. The difference was statistically significant with p-values < 0.05 ($p = 8.4 \times 10^{-14}$).

From the experimental results by the four specifications of the global elitism in Table 1, we can see that too many elite individuals as well as too few elite individuals



(a) 1 global elite individual
(b) 12 global elite individuals
(c) Cell-wise elitism
(d) Local elitism: Version 1
(e) Local elitism: Version 2
(f) Local elitism: Version 3

Fig. 2. Average performance of our cellular genetic algorithm over 100 runs for each setting

had negative effects on the performance of our cellular genetic algorithm. A possible negative effect of too many elite individuals is the decrease in the diversity of solutions. We show how individuals evolved in our cellular genetic algorithm in the two-dimensional objective space in Fig. 4 where all individuals in the initial, 20th, 50th and 10000th generation in a single run are depicted for each implementation scheme of elitism with the best neighborhood structures (see Table 1). In Fig. 4 (a) with the cell-wise elitism, all individuals converged on a single point before the 10000th generation. On the other hand, individuals of the 10000th generation had a certain amount of diversity in Fig. 4 (b) with local elitism Version 1. Such a diversity maintenance effect explains the best result by local elitism Version 1 in Table 1.

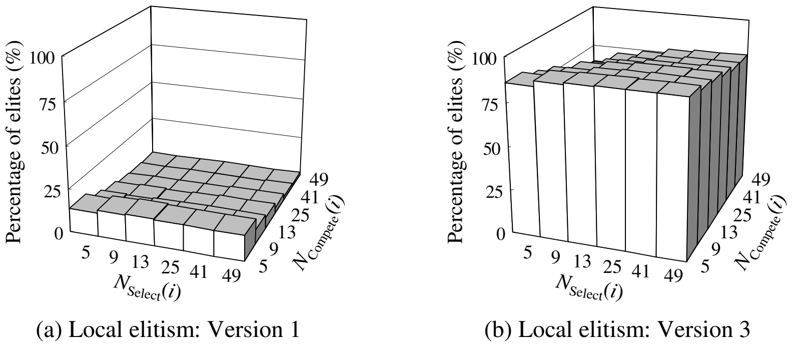


Fig. 3. Average percentage of elite individuals over 100 runs

Table 1. Experimental results with the best neighborhood structures (11×11 grid)

Elitism	Competition	Selection	Relative error
1 global elite individual	-	9	1.019 (0.124)
6 global elite individuals	-	49	0.461 (0.090)
12 global elite individuals	-	41	0.467 (0.099)
24 global elite individuals	-	41	0.479 (0.098)
Cell-wise elitism	-	25	0.524 (0.115)
Local Version 1	9	41	**0.410 (0.079)**
Local Version 2	9	41	0.436 (0.093)
Local Version 3	5	5	0.543 (0.128)

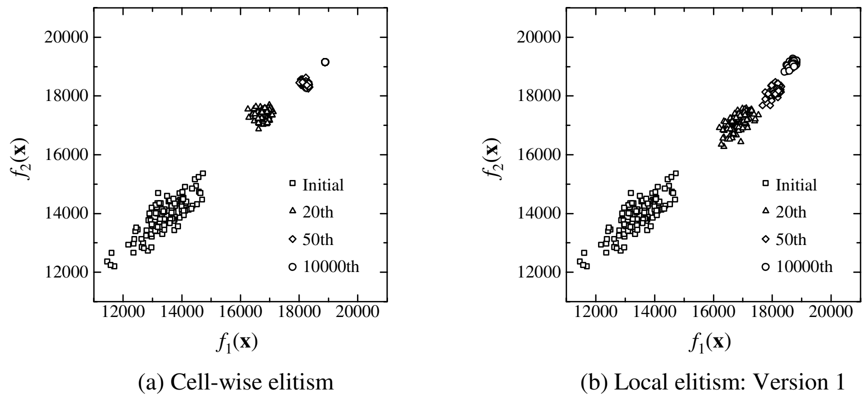


Fig. 4. Average percentage of elite individuals over 100 runs

One may think that the above-mentioned negative effect of the cell-wise elitism (where the replace-if-better policy was applied to all individuals) would be removed or remedied by the use of a larger grid-world. So we also performed computational experiments using a 21×21 grid-world with 441 cells. Experimental results are summarized in Table 2. Better results were obtained by increasing the number of elite individuals in Table 2. Whereas the difference in the average relative errors between the cell-wise elitism and local elitism Version 1 was decreased by the use of the larger grid-world in Table 2, the best result was still obtained by local elitism Version 1 with a small competition neighborhood and a large selection neighborhood.

Table 2. Experimental results with the best neighborhood structures (21×21 grid)

Elitism	Competition	Selection	Relative error
1 global elite individual	-	5	1.040 (0.122)
22 global elite individuals	-	41	0.299 (0.062)
44 global elite individuals	-	41	0.293 (0.070)
88 global elite individuals	-	25	0.293 (0.061)
Cell-wise elitism	-	9	0.289 (0.070)
Local Version 1	9	49	**0.269 (0.049)**
Local Version 2	5	5	0.280 (0.065)
Local Version 3	5	13	0.308 (0.074)

In order to examine the behavior of our cellular genetic algorithm on a multi-modal problem, we applied it to Schwefel function with 10 variables. Each variable was coded as a binary string of length 10 using gray coding. We used the same parameter specifications as in the previous computational experiments in Fig. 2. The mutation probability was specified as 1/100 (i.e., 1/(string length)). Experimental results are summarized in Table 3. The best result was obtained from the global elitism with only a single elite individual. We can also observe that the standard deviation was very large in Table 3 except for the best result case. This is because strong elitism prevented individuals from escaping from local optima. We can see from Tables 1-3 that the choice of an appropriate implementation of elitism is problem-dependent. For example, the cell-wise elitism was a bad choice in Table 3.

We also performed the same computational experiments using a larger mutation probability (0.05 instead of 0.01), a larger crossover probability (1.0 instead of 0.8) and a smaller computation load (500 instead of 2000 generations). In this case, we

Table 3. Experimental results on Schwefel function using the same parameter specifications as in the previous computational experiments on the 0/1 knapsack problem

Elitism	Competition	Selection	Relative error
1 global elite individual	-	5	**0.014 (0.100)**
6 global elite individuals	-	5	9.479 (32.14)
12 global elite individuals	-	5	14.53 (38.48)
24 global elite individuals	-	5	35.54 (68.05)
Cell-wise elitism	-	5	40.28 (71.51)
Local Version 1	41	5	1.529 (11.52)
Local Version 2	41	5	4.761 (23.21)
Local Version 3	13	5	55.68 (84.53)

needed strong elitism since the mutation probability was too large. Experimental results are summarized in Table 4 where the best result was obtained from the cell-wise elitism. From Table 4, we can see that the cell-wise elitism worked well when it was used together with high crossover and mutation probabilities.

Table 4. Experimental results on Schwefel function using different parameter specifications with a larger mutation probability, a larger crossover probability, and a less computation load

Elitism	Competition	Selection	Relative error
1 global elite individual	-	41	238.6 (129.9)
6 global elite individuals	-	25	84.56 (85.51)
12 global elite individuals	-	49	37.42 (60.90)
24 global elite individuals	-	25	23.23 (45.32)
Cell-wise elitism	-	9	**0.705 (0.402)**
Local Version 1	5	9	4.257 (1.963)
Local Version 2	5	9	5.246 (11.80)
Local Version 3	5	9	8.344 (23.21)

4 Conclusions

We examined various implementations of elitism through computational experiments using a cellular genetic algorithm with two neighborhood structures. It was demonstrated that the choice of an implementation scheme of elitism had a dominant effect on the performance of our cellular genetic algorithm. When we used the local elitism, the performance of our cellular genetic algorithm was sensitive to the specifications of the two neighborhood structures. The best result was obtained from Version 1 of the local elitism with a small competition neighbor and a large selection neighbor for a knapsack problem. In the global elitism and the cell-wise elitism, we used only a single neighborhood structure as in standard cellular algorithms. The performance of our cellular genetic algorithm was not sensitive to the specification of the neighborhood structure when we used the global elitism and the cell-wise elitism. We also demonstrated that the cell-wise elitism was not always a good choice whereas it has been often used in many studies on cellular genetic algorithms. This work was partially supported by Grant-in-Aid for Scientific Research (B): KAKENHI (20300084).

References

1. Alba, E., Dorronsoro, B.: The Exploration/Exploitation Tradeoff in Dynamic Cellular Genetic Algorithms. IEEE Trans. on Evolutionary Computation 9, 126–142 (2005)
2. Alba, E., Tomassini, M.: Parallelism and Evolutionary Algorithms. IEEE Trans. on Evolutionary Computation 6, 443–462 (2002)
3. Alba, E., Troya, J.M.: Cellular Evolutionary Algorithms: Evaluating the Influence of Ratio. In: Parallel Problem Solving from Nature - PPSN VI. LNCS, vol. 1917, pp. 29–38. Springer, Berlin (2000)
4. Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Springer, Berlin (2000)

5. Charlesworth, B.: A Note on the Evolution of Altruism in Structured Demes. The American Naturalist 113, 601–605 (1979)
6. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
7. Giacobini, M., Tomassini, M., Tettamanzi, A.G.B., Alba, E.: Selection Intensity in Cellular Evolutionary Algorithms for Regular Lattices. IEEE Trans. on Evolutionary Computation 9, 489–505 (2005)
8. Gorges-Schleuter, M.: ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy. In: Proc. of 3rd International Conference on Genetic Algorithms, pp. 422–427 (1989)
9. Gorges-Schleuter, M.: A Comparative Study on Global and Local Selection in Evolutionary Strategies. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 367–377. Springer, Heidelberg (1998)
10. Ifti, M., Killingback, T., Doebelic, M.: Effects of Neighborhood Size and Connectivity on the Spatial Continuous Prisoner's Dilemma. Journal of Theoretical Biology 231, 97–106 (2004)
11. Ishibuchi, H., Doi, T., Nojima, Y.: Effects of Using Two Neighborhood Structures in Cellular Genetic Algorithms for Function Optimization. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 949–958. Springer, Heidelberg (2006)
12. Ishibuchi, H., Namikawa, N.: Evolution of Iterated Prisoner's Dilemma Game Strategies in Structured Demes under Random Pairing in Game Playing. IEEE Trans. on Evolutionary Computation 9, 552–561 (2005)
13. Jagerskupper, J., Storch, T.: How Comma Selection Helps with the Escape from Local Optima. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 52–61. Springer, Heidelberg (2006)
14. Manderick, B., Spiessens, P.: Fine-Grained Parallel Genetic Algorithms. In: Proc. of 3rd International Conference on Genetic Algorithms, pp. 428–433 (1989)
15. Sarma, J., De Jong, K.: An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 236–244. Springer, Heidelberg (1996)
16. Sarma, J., De Jong, K.: An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. In: Proc. of 7th International Conference on Genetic Algorithms, pp. 181–186 (1997)
17. Slatkin, M., Wilson, D.S.: Coevolution in Structured Demes. Proc. of National Academy of Sciences 76, 2084–2087 (1979)
18. Spiessens, P., Manderick, B.: A Massively Parallel Genetic Algorithm: Implementation and First Analysis. In: Proc. of 4th International Conference on Genetic Algorithms, pp. 279–286 (1991)
19. Whitley, D.: Cellular Genetic Algorithms. In: Proc. of 5th International Conference on Genetic Algorithms, p. 658 (1993)
20. Wilson, D.S.: Structured Demes and the Evolution of Group-Advantageous Traits. The American Naturalist 111, 157–185 (1977)
21. Wilson, D.S.: Structured Demes and Trait-Group Variation. The American Naturalist 113, 606–610 (1979)
22. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans. on Evolutionary Computation 3, 257–271 (1999)

The Generalisation Ability of a Selection Architecture for Genetic Programming

David Jackson

Dept. of Computer Science, University of Liverpool
Liverpool L69 3BX, United Kingdom
djackson@liverpool.ac.uk

Abstract. As an alternative to various existing approaches to incorporating modular decomposition and reuse in genetic programming (GP), we have proposed a new method for hierarchical evolution. Based on a division of the problem's test case inputs into subsets, it employs a program structure that we refer to as a selection architecture. Although the performance of GP systems based on this architecture has been shown to be superior to that of conventional systems, the nature of evolved programs is radically different, leading to speculation as to how well such programs may generalise to deal with previously unseen inputs. We have therefore performed additional experimentation to evaluate the approach's generalisation ability, and have found that it seems to stand up well against standard GP in this regard.

1 Introduction

In recent years, there has been much research activity aimed at the question of how to scale genetic programming (GP) to deal with complex, high-level problems. One of the most promising avenues of research is that of employing decomposition and reuse techniques to break a task down into more manageable, easily solvable sub-systems which can then be combined to create solutions to the original problem. Among the various approaches to this are Koza's Automatically Defined Functions (ADFs) [1,2], Angeline and Pollack's Module Acquisition technique [3], Rosca and Ballard's Adaptive Representation through Learning [4], and Walker and Miller's work on module encapsulation in Cartesian GP [5].

Most of these existing methods involve the identification and encapsulation of useful code fragments as they arise during the evolutionary process. An alternative approach, which has been the subject of our own investigations, is to specify the goals of the subsystems in advance, and then to evolve solutions to each of those goals in independently executing processes. There are various ways of performing such goal-directed hierarchical learning in GP, but the one that we focus on here involves the use of what we call a selection architecture. This architecture is based on the idea that each subsystem be made responsible for dealing with only a subset of the range of possible inputs to the program as a whole. The subsystems are encoded as branches of the main program, connected together by a root node which is given the role of activating the appropriate branch for each input received. In this way, the problem is reduced to a number of greatly simplified and independent subtasks that are often trivial to solve.

As we shall see later, the performance of this selection architecture is substantially better than that of more conventional GP systems. However, the nature of programs based on this architecture is radically different from that of programs evolved by more standard GP techniques. Although the performance improvements are certainly an advantage, it has to be asked whether there is a trade-off against other properties of the programs. In particular, one concern is how well such individuals are capable of generalising.

In assessing generalisation ability, it is suggested that, at the very least, input cases should be divided into two sets: a training set and a test set [6,7]. The training set is used during the evolutionary process itself, while the test set contains input values that have not been previously encountered. Some researchers go further than this two-stage mechanism by advocating the addition of an intermediate validation set to allow programs to gauge their generalisation ability during evolution [8].

Various suggestions have been made as to how best to improve generalisation in GP. Francone, Nordin and Banzhaf, for example, have discussed how to benchmark and analyse the generalisation capability of their Compiling Genetic Programming System [9], and have found that it is significantly affected by the rate of mutation used [10]. Vanneschi et al [11] have found that Pareto multi-optimisation has a marked positive impact, whilst Droste [12] argues that generalisation can be promoted in Boolean functions through the use of a program representation known as ordered binary decision diagrams.

There is evidence to suggest that generalisation tends to be better in smaller programs, the explanation being that compactness requires constructs capable of dealing with multiple cases [13]. For this reason, some researchers have argued for the use of parsimony pressure to drive down progam size whilst driving up generalisation prowess [8]. However, it has also been shown that, in some circumstances, too strong a bias towards low complexity can increase generalisation error [14].

We shall return to this generalisation issue as it applies specifically to our selection architecture in Section 3, and will go on to present some experiments which assess its generalisation capabilities in relation to those of standard GP. Before we can do that, we need to present more details concerning the architecture itself.

2 A Selection Architecture

In most conventional GP systems, the program code of individuals is represented in a tree structure, the internal nodes being members of the function set, and the leaf nodes being taken from the terminal set. Alternatives to this format include linear code and more general graph structures. Attempts to evolve code that is more modular in nature, and which makes use of evolving sub-structures, involve modifications of these basic forms. The best known approach is that of Koza's Automatically Defined Functions (ADFs), which evolve a number of function-defining branches in tandem with a main branch that may invoke these functions.

The architecture we propose is also hierarchical in nature, but unlike ADFs it assigns a more definite purpose to each of the lower-level subsystems. Conceptually, it is quite simple, as shown in Figure 1. Like an ADF tree, this structure also contains a pre-defined number of branches off the root node. Unlike its ADF counterpart, however, there is no 'main' branch. Instead, each branch is charged with the responsibility of handling a subset of the input test cases to be applied to the individual as a whole.

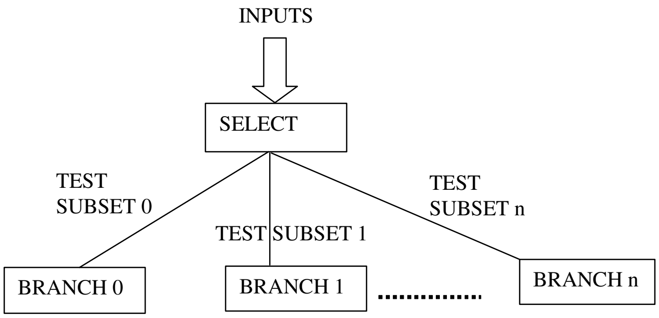


Fig. 1. Selection architecture for test-subset approach

The idea is that, given a set or range of input cases, we partition it into a number of subsets. Code for handling each subset is then evolved independently in separate branches of the program. Decomposition of the original problem in this way should lead to a number of sub-objectives which, in isolation, are easier to solve via evolutionary computation. A trade-off is an increase in the number of code fragments that must be evolved to solve all branches.

The branches in the selection architecture are not functions in the ADF sense: they are simply code fragments composed from the normal terminal and function sets of the problem. Despite this, each branch does not interact directly with other branches; rather, it is evolved separately and independently. This is a key difference from the ADF architecture, in which all branches evolve simultaneously towards the solution of a problem and the evolutionary value of each branch is judged according to the contribution it makes to the fitness of the individual as a whole. In the selection architecture each branch has its own evolutionary target, its fitness being calculated according to how well it deals with its assigned subset of test cases. Evolutionary effort is focused on one branch at a time rather than all branches at once, although the independent nature of the code fragments means that all branches could readily be evolved in parallel on a multiprocessor machine.

A further difference between the two architectures is that, whereas the number of branches in an ADF system is rather arbitrary, the branch count in the selection architecture is determined by the number of subsets into which the test cases have been divided. These branches are linked together at a single root node, the purpose of which is to decide which branch to activate for a particular combination of inputs. The root node therefore acts as a kind of switch, its exact form depending on the problem being solved and the language being used to encode evolved programs. In most situations it will correspond to a straightforward 'case' statement or a nested 'if-then-else' construct.

Table 1. Performance comparisons for the even-5 parity problem

Approach	**Success rate (%)**	**Comp. Effort**
Standard GP	0	-
ADF GP	32	864,000
4 branches, 8 cases each	91	192,000
8 branches, 4 cases each	100	16,000

An extensive assessment of the performance of this architecture in comparison with other methods has been given elsewhere [15]. A single example is the even-5 parity problem, in which the aim is to evolve a Boolean design that returns a TRUE output if the number of logic one values on its 5 inputs D0-D4 is even, FALSE otherwise. In making the comparisons we use two forms of the selection architecture: one with 4 branches, each branch dealing with 8 of the 32 possible input cases, and one with 8 branches, each responsible for 4 input cases. Table 1 shows how these systems fare against standard GP and a GP system using ADFs (detailed problem parameters are given in [15]). The figures given measure the success rate at finding solutions over 100 runs, and also the computational effort statistic as defined by Koza [1].

Like Koza, we found that discovering a solution to the even-5 parity problem using standard GP is extremely difficult. By incorporating an ADF mechanism we were able to get much better results, with a success rate of 32%. When we try the selection architecture using 4 branches, the success rate is almost triple that achieved in the ADF system, leading to a huge decrease in the computational effort. As before, the use of 8 branches gives us a solution on every run, and an associated computational effort that is comparatively tiny.

3 The Generalisation Issue

We have seen in the previous section that the use of a selection architecture as a basis for genetic programming can lead to dramatic improvements in performance, measured in a variety of ways. However, the nature of programs evolved within this framework is very different from that of programs generated using more conventional GP approaches. Whereas standard GP attempts to derive a program capable of matching all of a set of inputs to the corresponding outputs, the selection approach attempts to evolve a collection of related sub-systems, each of which is responsible for a subset of the possible inputs. To illustrate this, consider the even-4 parity problem. A solution generated using a 4-branch selection architecture is:

```
   SWITCH (INT(D3..D0))
   CASE 0..3:
   NAND(OR(D0 D1) NAND(D0 D1))
   CASE 4..7:
   NOR(NOR(OR(D1 D2) OR(D1 D1)) NAND(OR(D0 D1) NAND(D0 D1)))
   CASE 8..11:
   AND(AND(OR(NAND(OR(D1 D0) AND(D1 D1)) OR(NAND(D1 D0) NAND(D3 D1)))
OR(NOR(NOR(D2 D0) NAND(D2 D2)) OR(D1 D0))) OR(NAND(NOR(D2 NOR(D0 D0))
NOR(NAND(D0 D0) NOR(D0 D2))) NAND(NAND(AND(D1 D2) NOR(D0 D3)) AND(AND(D2
D3) NAND(D3 D1)))))
   CASE 12..15:
   OR(AND(AND(NAND(AND(D1 D0) OR(D2 D1)) AND(NAND(D0 D3) NAND(D1 D2)))
AND(OR(AND(D2 D1) NAND(D3 D0)) OR(NAND(D1 D3) OR(D2 D3)))) AND(D1 D0))
```

In this solution, the root node is implemented as a form of case statement, acting on an integer representation of the four binary inputs and directing control flow to the appropriate branch for that value. Each branch has evolved to deal with its own particular subset of values, and not for any other. As such, a given branch may be quite trivial, such as the first branch of our solution above, in which the terminals D2 and D3 do not even appear. Whilst this may make it quicker and easier to evolve code for such branches, it also gives rise to questions as to whether, in the general case, they possess sufficient complexity to deal with inputs that do not form part of the training set.

The point is further illustrated with a symbolic regression problem. Here, the aim is to evolve a formula to fit a number of x-y data pairs. In our experiments, the formula is the polynomial $4x^4 - 3x^3 + 2x^2 - x$. When plotted as graphs, the differences between a conventional solution and a selection-based solution are not readily discernible. However, for 'poor' programs which do not attain all hits, the differences are a lot more visually apparent. Figure 2 shows the behaviour of a best-of-run program obtained using conventional GP for finding the specified polynomial. It achieves just 12 out of 32 hits and has a fitness value of 3.78.

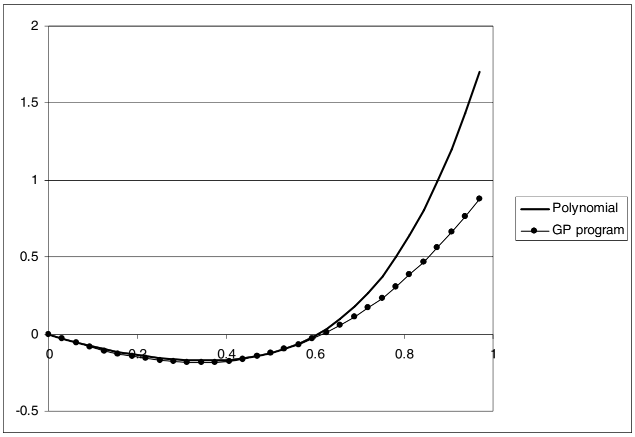


Fig. 2. Behaviour of a program evolved by standard GP for the symbolic regression problem

It can be seen that the Y-values produced by the GP program form a smooth curve. This curve fits to the polynomial quite closely in the first two thirds of the graph, and then begins to deviate from it. By way of contrast, consider the graph of Figure 3, which is for a 4-branch selection-based program evolved as the best of a run.

This program has more hits (16) than the conventional GP program above, and a much better fitness (1.2). However, the division of labour into four separate sub-systems is clear, with the upper two finding very poor linear approximations to the polynomial curve. Again, it must be wondered whether evolved programs of this nature can hope to compete with regular GP in being able to generalise to cope with previously unseen input values.

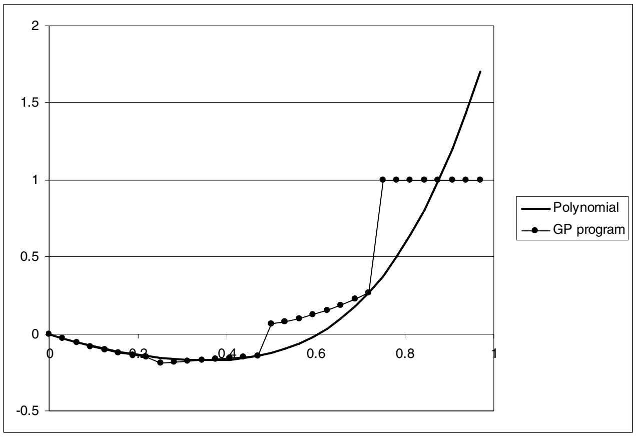


Fig. 3. Behaviour of a program evolved by selection GP for the symbolic regression problem

4 Generalisation Experiments

We begin with the even-4 parity problem, the standard GP parameters for which are set out in Table 2.

Table 2. GP parameters for the even-4 parity problem

Objective	To evolve a program capable of determining if the number of logic 1s on the 4 inputs is even
Terminal set	D0, D1, D2, D3
Function set	AND, OR, NAND, NOR
Initial population	Ramped half-and-half
Evolutionary process	Steady-state; 5-candidate tournament selection
Fitness cases	16, representing all combinations of inputs
Fitness	Number of mismatches with expected outputs (0-16)
Success predicate	Zero fitness (solution found)
Other parameters	Pop size=500; Gens=51; prob. crossover=0.9; no mutation; prob. internal node used as crossover point=0.9

In trying to assess the generalisation capabilities of our systems, we divide the fitness cases into two sets: a training set TR and a test set TE. Programs are evolved using the training set alone, and then the best program at the end of each run is applied to the test set to determine performance on these new values. For the even-4 parity problem, which has 16 test input cases in all, we perform the experiments with a test set of size 4 and then with a test set of size 8.

In deciding which input case belongs to which set, we use the following algorithm: divide the input range into a number of partitions equal to the size of the test set, then form the test set by selecting one value at random from each partition. For example, in our even-4 parity problem, the possible values on the four binary inputs correspond to the decimal integers 0-15. If we require a test set of size 4, we partition our input range equally to give 0-3, 4-7, 8-11 and 12-15. We then choose one input value at random from each partition, so that our final test set might be the input values {2,5,11,14}. Using this simple algorithm allows a certain degree of randomness whilst at the same time ensuring that programs based on the selection architecture end up with an equal number of input cases per branch, irrespective of the sizes of the test and training sets.

Table 3 shows the results obtained when comparing standard GP against 2-branch and 4-branch forms of the selection architecture for the even-4 parity problem, using two different test set sizes.

Table 3. Comparisons of generalisation ability for the even-4 parity problem (* indicates no statistical difference)

	TE size 4 (TR size 12)		**TE size 8 (TR size 8)**	
	Train score	*Test score*	*Train score*	*Test score*
Standard GP	0.23	3.96	0.00	7.56
2 branch	0.03	3.85	0.00(*)	7.58(*)
4 branch	0.00	3.96(*)	0.00(*)	7.62(*)

The scores in this table are averages of fitness values obtained over 100 runs. Fitness is computed as the number of mismatches with expected outputs, and so lower values indicate higher fitness, zero being the ideal. The results from all runs were also tested for statistical significance via the use of a t-test with an alpha value of 0.05. Results for the selection architecture that are not significantly different from the standard GP results are marked in the table by an asterisk.

When a training set of 12 values was used (the remaining 4 values forming the test set), the average score obtained by standard GP during that training was 0.23. This is bettered by both the 2-branch and 4-branch versions of the selection architecture, the latter finding programs which dealt with every training case on all runs. During the testing phase, the 2-branch selection system again fared better, while the 4-branch version achieved the same score. When the training set was reduced and the test set increased to 8 values, all systems found programs in each run that could deal with the training inputs. The results of the test phase did not differ by a statistically significant amount in the various systems.

Overall, then, it can be said that the generalisation ability of the selection architecture is at least as good as that of standard GP for the even-4 parity problem.

In the 5-input version of the even parity problem, the parameters are almost identical to those presented in Table 2, except that the population size is increased to 2000, and the total number of fitness cases is doubled to 32 because of the additional input. Once again, standard GP is compared with selection-based GP, this time using 2, 4 and 8 branches. The results for three different TE and TR sizes are given in Table 4.

Table 4. Comparisons of generalisation ability for the even-parity problem

	TE size 4 (TR size 28)		**TE size 8 (TR size 24)**		**TE size 16 (TR size 16)**	
	Train score	*Test score*	*Train score*	*Test score*	*Train score*	*Test score*
Standard GP	3.28	3.83	1.96	7.57	0.56	15.02
2 branch	1.69	3.87(*)	0.35	7.81	0.00	15.54
4 branch	0.11	3.89(*)	0.00	7.72(*)	0.00	15.31
8 branch	0.00	3.97	0.00	7.92	0.00	15.42

Once again, the scores obtained by the selection architecture during training are substantially better than those for standard GP during that phase. However, to a lesser extent, the situation is reversed during the subsequent test phase. When the size of the test set TE is only 4, there is little significant difference. For larger TE sizes the scores for selection-based GP tend to be worse than the conventional counterpart. These differences are marginal in absolute terms but, on the whole, they are statistically significant.

For the symbolic regression problem, the GP parameters are as given in Table 5. The range of x values for this problem is specified as [0.0,1.0). In principle, any number of data points could be chosen from this range in order to evaluate fitness; we chose 32 to make it correspond more directly to the even-5 parity problem in terms of TE and TR sizes and the number of selection branches.

The comparative results for this problem are given in Table 6. A point to note here is that the scores given represent the number of 'hits,' where a hit indicates that the evolved expression produces a value that is within an acceptable deviation limit from the expected polynomial value. As such, higher scores equate to better fitness, unlike the parity problems discussed above.

Table 5. GP parameters for the polynomial symbolic regression problem

Objective	Symbolic regression of the polynomial $4x^4 – 3x^3 + 2x^2 - x$
Terminal set	x
Function set	+, -, *, / (division protected to return large value on divide-by-zero)
Initial population	Ramped half-and-half
Evolutionary process	Steady-state; 5-candidate tournament selection
Fitness cases	32 x-values in the range [0, 1), from 0.0 increasing in steps of 1/32, plus corresponding y values
Fitness	Sum of absolute errors in calculated y values
Success predicate	32 hits, a hit being error less than 0.01
Other parameters	Pop size=500; Gens=51; prob. crossover=0.9; no mutation; prob. internal node used as crossover point=0.9

With regard to training, the pattern is similar to that seen earlier, with the selection architecture performing much better than conventional GP in discovering programs capable of dealing with the training data. For the testing phase, however, the situation is somewhat less unequivocal. With a TE size of 4 or 8, some of the selection-based GP test results are better, some worse, and some not statistically distinct from those of standard GP. Only when the TE size is increased to 16 do all forms of the selection architecture exhibit higher scores, although even then one of those scores is not statistically significant.

Table 6. Comparisons of generalisation ability for the symbolic regression problem

	TE size 4 (TR size 28)		**TE size 8 (TR size 24)**		**TE size 16 (TR size 16)**	
	Train score	*Test score*	*Train score*	*Test score*	*Train score*	*Test score*
Standard GP	17.58	2.76	14.08	4.94	9.34	9.26
2 branch	23.22	2.86(*)	18.66	5.56	12.92	11.82
4 branch	22.74	2.12	21.22	5.36(*)	12.94	10.56
8 branch	22.00	2.24	21.8	4.06	14.6	10.18(*)

5 Conclusions

This work has provided further evidence to support the claim that, in terms of the ability to discover programs that can produce the correct outputs for a known set of outputs, the performance of a selection architecture for GP is substantially superior to that of more conventional systems. With regard to the comparative generalisation ability of these systems, however, the results are less conclusive. On the basis of the admittedly limited set of experiments described here, it would not seem unreasonable to claim that selection-based GP is no worse than standard GP in this regard. As the selection-based method matures and is applied to more problem domains, it is hoped to gather further evidence to support this claim. What is certainly true of both standard and selection-based GP, however, is that neither exhibits generalisation powers that could be said to be impressive. This is especially so for the even parity problems, where the level of training appears to make little difference to the outcome: in almost every run of every system, the number of test cases failed is remarkably close to the size of the test set! Clearly, a good deal of additional research is required into this area, which is key to the future success and acceptability of GP as a solution-finding tool.

References

1. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
2. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge (1994)

3. Angeline, P.J., Pollack, J.: Evolutionary Module Acquisition. In: Proc. 2nd Annual Conf. on Evolutionary Programming, La Jolla, CA, pp. 154–163 (1993)
4. Rosca, J.P., Ballard, D.H.: Discovery of Subroutines in Genetic Programming. In: Angeline, P., Kinnear Jr., K.E. (eds.) Advances in Genetic Programming 2, ch. 9, pp. 177–202. MIT Press, Cambridge (1996)
5. Walker, J.A., Miller, J.F.: Evolution and Acquisition of Modules in Cartesian Genetic Programming. In: Keijzer, M., O'Reilly, U.-M., Lucas, S.M., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 187–197. Springer, Heidelberg (2004)
6. Kushchu, I.: Genetic Programming and Evolutionary Generalization. IEEE Transactions on Evolutionary Computation 6(5), 431–442 (2002)
7. Kushchu, I.: An Evaluation of Evolutionary Generalisation in Genetic Programming. Artificial Intelligence Review 18, 3–14 (2002)
8. Gagné, C., Schoenauer, M., Parizeau, M., Tomassini, M.: Genetic Programming, Validation Sets and Parsimony Pressure. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 109–120. Springer, Heidelberg (2006)
9. Francone, F.D., Nordin, P., Banzhaf, W.: Benchmarking the Generalization Capabilities of a Compiling Genetic Programming System Using Sparse Data Sets. In: Koza, J.R., et al. (eds.) Proc. 1st Annual Genetic Programming Conf (GP 1996), pp. 72–80. MIT Press, Cambridge (1996)
10. Banzhaf, W., Francone, F.D., Nordin, P.: The Effect of Extensive Use of the Mutation Operator on Generalization in Genetic Programming using Sparse Data Sets. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 300–309. Springer, Heidelberg (1996)
11. Vanneschi, L., Rochat, D., Tomassini, M.: Multi-Optimization Improves Genetic Programming Generalization Ability. In: Thierens, D., et al. (eds.) Proc. GECCO 2007, p. 1759. ACM Press, New York (2007)
12. Droste, S.: Efficient Genetic Programming for Finding Good Generalizing Boolean Functions. In: Koza, J.R., et al. (eds.) Proc. 2nd Annual Conf. on Genetic Programming, pp. 82–87. Morgan Kaufmann, San Francisco (1997)
13. Rosca, J.: Generality versus Size in Genetic Programming. In: Koza, J.R., et al. (eds.) Proc. 1st Annual Genetic Programming Conf (GP 1996), pp. 381–387. MIT Press, Cambridge (1996)
14. Cavaretta, M.J., Chellapilla, K.: Data Mining Using Genetic Programming: The Implications of Parsimony on Generalization Error. In: Proc. CEC 1999, Washington, DC, USA, pp. 1330–1337 (1999)
15. Jackson, D.: The Performance of a Selection Architecture for Genetic Programming. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 170–181. Springer, Heidelberg (2008)

Reinforcement Learning: Insights from Interesting Failures in Parameter Selection

Wolfgang Konen and Thomas Bartz–Beielstein

Cologne University of Applied Sciences,
Faculty for Computer Science and Engineering Science,
51643 Gummersbach, Germany
{wolfgang.konen|thomas.bartz-beielstein}@fh-koeln.de

Abstract. We investigate reinforcement learning methods, namely the temporal difference learning TD(λ) algorithm, on game-learning tasks. Small modifications in algorithm setup and parameter choice can have significant impact on success or failure to learn. We demonstrate that small differences in input features influence significantly the learning process. By selecting the right feature set we found good results within only 1/100 of the learning steps reported in the literature. Different metrics for measuring success in a reproducible manner are developed. We discuss why linear output functions are often preferable compared to sigmoid output functions.

1 Introduction

Reinforcement learning (RL) is a powerful optimization technique in situations where a learning agent does not receive a direct target signal for each (observation, decision) pair. The agent receives only a reward from the environment and does not learn a target output function. Often the reward is only given after a sequence of decisions has been taken. Reinforcement learning attempts to mimic one major way how animals or humans learn in natural environments. Instead of being told what to do, they learn through experience. In a similar way, reinforcement learning agents learn to interact with an unknown and unspecified environment.

Sutton's well-known *temporal difference* (TD) learning algorithm is a specific method to deal with the credit assignment problem in control and decision tasks [1]. Based on this work, Tesauro designed in 1994 the famous TD-Gammon agent which learned basically from self-play how to play the game of backgammon at world champion level [2]. This made TD learning very popular, and many successful applications have been reported since then. However, numerous researchers also tried to apply TD (or RL in general) to distinct problems and found quite mixed results in terms of convergence speed and/or decision quality of the learning agent. Despite of elegance of RL theory and the simplicity of the basic TD ideas, the implementation of the algorithms is not trivial: tiny implementation details can decide about complete success or failure.

We study in this paper the application of TD learning to simple game-play tasks as a preparation for more complex learning tasks. We are interested in elements of the algorithm which have significant impact on convergence speed and/or success or failure of the learning agent. A better understanding of surprising failures on simple tasks might help to configure algorithms in the right way for more complex tasks.

In Sect. 2 we describe the TD algorithm and its application to the game-learning tasks. In Sect. 3 we describe our metrics for measuring the quality of the learning agent and present our results, which are further discussed in terms of general insights in Sect. 4.

2 Methods

We consider two simple games:

Nim-3. A simplified variant of the game Nim, where N tokens are on the table, each player can take 1, 2, or 3 tokens and the winner is the one who takes the last token. The state space has $2N$ states. The optimal strategy is to leave $3+1$ tokens for the opponent. Although almost trivial, we are interested in situations where RL fails to learn the task or is considerably slow in learning it.

TicTacToe. The board contains 3×3 fields, each player in each move marks (with `X` or `O`) a field and the winner is who gets "3 in a row" (horizontal, vertical, diagonal). The state space contains 5478 states. This is small enough that a standard minimax agent can perform exhaustive search for each state and find the best move.

A state in strategic games is usually described by the current board position and the player who made the last move (so-called *after state* [3]). An example for TicTacToe is shown in Fig. 1. Following the ideas of Tesauro [2], the RL agent learns the game function $V(\boldsymbol{s}_t)$, which ideally gives for each after state the probability that player $p = +1$, i.e., "`X`" will win. Given a certain board position, the strategy for player $p = +1$ is to select the next move which maximizes $V(\boldsymbol{s}_{t+1})$, while player $p = -1$ ("`O`") tries to minimize $V(\boldsymbol{s}_{t+1})$. A state can be encoded by collecting row-by-row the board positions into a state vector with $+1$ for each "`X`", 0 for each unoccupied field and -1 for each "`O`". Together with

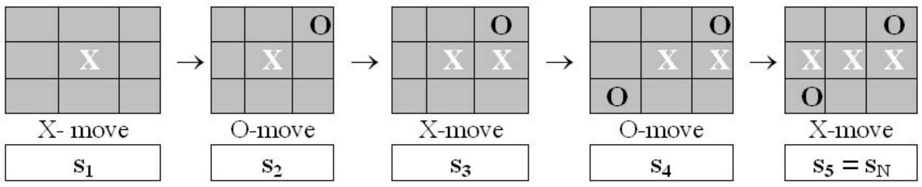


Fig. 1. Some after states for the game TicTacToe

the player who made the last move we get for example in Fig. 1 the following state representation for state $\boldsymbol{s}_4$, which is a safe win for player "`X`":

$$\boldsymbol{s}_4 = \{00\text{-}1, 011, \text{-}100, -1\}, \qquad V(\boldsymbol{s}_4) = 1.000 \tag{1}$$

Even for moderate games the state space is usually too large to be represented as a table and it is impossible to visit every state sufficiently often during learning. To overcome this problem, a function approximation scheme is used where each state $\boldsymbol{s}$ is transformed into a feature state $\boldsymbol{g}(\boldsymbol{s})$ and the function $f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}))$ with internal parameter vector $\boldsymbol{w}$ (the weight vector) approximates $V(\boldsymbol{s})$.

The TD algorithm aims at learning the function $f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}))$. It does so by setting up an (initially inexperienced) RL agent who plays a sequence of games against himself. It learns from the environment which gives a reward $r \in \{0.0, 0.5, 1.0\}$ for { `O`-win, tie, `X`-win } at the end of each game. The main ingredient is the *temporal difference* (TD) error signal

$$\delta_t = r_{t+1} + \gamma V(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t), \tag{2}$$

where r_{t+1} is the reward for state $\boldsymbol{s}_{t+1}$ (0 in a rewardless state, the game reward r when $t+1$ is the final state) and $V(\boldsymbol{s}_{t+1})$ is the game value for $\boldsymbol{s}_{t+1}$. The idea is to remember from state $\boldsymbol{s}_t$ the value $V(\boldsymbol{s}_t)$ and the gradient $\nabla_w f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}_t))$ of the function f with respect to the weights $\boldsymbol{w}$, to wait for the next state $\boldsymbol{s}_{t+1}$, and to apply then a learn step for the former state $\boldsymbol{s}_t$. Thus the error signal aims at bringing the game value $V(\boldsymbol{s}_t)$ closer to the (best) successor game value $\gamma V(\boldsymbol{s}_{t+1})$ in a rewardless state or closer to the sum $r_{t+1} + \gamma V(\boldsymbol{s}_{t+1})$ in a final state. The discount parameter γ is usually close to 1.

Typical approximation functions are

– A linear function $f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s})) = \boldsymbol{w} \cdot \boldsymbol{g}(\boldsymbol{s})$ (or the sigmoid of this linear function)
– A backpropagation net with weights $\boldsymbol{w}$ and input $\boldsymbol{g}(\boldsymbol{s})$.

In both cases the learning step uses a variant of gradient descent with the so-called eligibility vectors $\boldsymbol{e}_t$. The core of the TD(λ)-algorithm is given as pseudo code as Algorithm 1. After the network is initialised with random weights, Algorithm 1 is called for G games to produce a trained RL agent. Usually the learning parameter α and the exploration parameter ϵ are slowly decreased in the sequence of the games, e.g., α decreases exponentially from α_{init} to α_{final}.

For each of the games Nim-3 and TicTacToe we explore different feature sets which are defined in Tab. 1. As an illustration consider TicTacToe state $\boldsymbol{s}_4$ in Fig. 1, which gives rise to the following feature vectors in the sets $T1$ and $T3$, resp.:

$$T1 : \boldsymbol{g}(\boldsymbol{s}_4) = (3, 0, 0, 2, 1, 0)$$
$$T3 : \boldsymbol{g}(\boldsymbol{s}_4) = (3, 0,\ \ 2, 1,\ \ 3, 2,\ \ 0, 0,\ \ 1, 0,\ \ 0, 0, -1,\ \ 0, 1, 1,\ \ -1, 0, 0)$$

Note that there is only a small difference between $F0$ and $F2$ (the 1 is replaced by p), but this has a large impact on learning, as we will see below.

Algorithm 1. "Self-play": Incremental TD(λ)-algorithm for strategic games

Input: player p_0 [=+1 ("X") or -1 ("O")] for the first move, initial state $\boldsymbol{s}_0$, and a (partially trained) function $f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}_t))$ to calculate the game function $V(\boldsymbol{s}_t)$.

```
 1: $V_{\text{old}} := f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}_0))$ and $t := 0$        ▷ with player $-p_0$ in after state $\boldsymbol{s}_0$
 2: $\boldsymbol{e}_0 := \nabla_{\boldsymbol{w}} f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}_0))$
 3: for ($p := p_0$; 1; $-p \to p$, $t++$) do        ▷ switch forever between players
 4:     select random number $r \in [0, 1]$
 5:     if $r < \epsilon$ then select randomly an after state $\boldsymbol{s}_{t+1}$        ▷ explorative move
 6:     else select after state $\boldsymbol{s}_{t+1}$ which maximizes $p \cdot f(\boldsymbol{w}; \boldsymbol{s}_{t+1})$        ▷ greedy move
 7:     get response $V(\boldsymbol{s}_{t+1}) := f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}_{t+1}))$ and reward $r_{t+1} := r(\boldsymbol{s}_{t+1})$ from environment
 8:     calculate error signal $\delta_t := r_{t+1} + \gamma V(\boldsymbol{s}_{t+1}) - V_{\text{old}}$
 9:     if $\boldsymbol{s}_{t+1}$ is greedy move or $\boldsymbol{s}_{t+1}$ is final state then
10:         make learn step $\boldsymbol{w} := \boldsymbol{w} + \alpha \delta_t \boldsymbol{e}_t$
11:     end if
12:     if $\boldsymbol{s}_{t+1}$ is final state then break        ▷ exit for-loop
13:     $V_{\text{old}} := y_{t+1} := f(\boldsymbol{w}; \boldsymbol{g}(s_{t+1}))$        ▷ because $\boldsymbol{w}$ has changed!
14:     $\boldsymbol{e}_{t+1} := \gamma\lambda \boldsymbol{e}_t + \nabla_{\boldsymbol{w}} f(\boldsymbol{w}; \boldsymbol{g}(\boldsymbol{s}_{t+1}))$        ▷ becomes $\boldsymbol{e}_t$ for the next iteration
15: end for
```

3 Evaluation

We measure the success of a trained RL agent by different metrics.

Nim-3. The $2N$ possible outcomes of the game function V can be directly evaluated. If the value of $p \cdot V$ in a after state for player p with $s = 4m$ tokens, $m = 0, 1, 2, \ldots$, is larger than in the states with $s+2, s+1, s-1, s-2$, then the agent will play optimally on all possible moves and we term such an agent a *success*.

TicTacToe. We evaluate in a set of 40 selected states whether the RL agent produces the same move as the optimal move suggested by the minimax agent (or produces an equivalent move having the same minimax score). The *percentage of correct moves* is a measure of success. This metric explores the state space if the 40 selected states cover relevant aspects of the state space.

TicTacToe. In a *tournament*, the RL agent plays 500 games against other agents, either as player X (the starting player) or as player O. We measure the percentage of X-wins, ties, and O-wins. The success rate in a tournament is the simplest and, at the end of the day, most relevant metric. But, note that a tournament against the minimax agent will produce always the same moves and thus will explore only a tiny fraction of the state space.

The success rate in the Nim-3 metric is measured as the average over 500 realisations of the RL agent. The results in Fig. 2 show the following: The earlier each curve rises to 1.0 the faster the RL agent has learned the concept. It is clearly seen that learning is faster without a sigmoid on the output and that the linear net learns considerably faster than its backpropagation companion. All runs use the feature set $F0$, which is simpler to learn than $F1$ or $F2$.

Table 1. Feature sets for Nim-3 and TicTacToe. Each feature vector is an M-dimensional vector: $(f_0, \ldots, f_{M-1})$ for Nim-3 and $(t_0, \ldots, t_{M-1})$ for TicTacToe. Singlets in TicTacToe are lines (horizontal, diagonal, vertical) with exactly one token of player p, the rest of the fields being empty; similar for doublets and triplets. A crosspoint is an empty field belonging to at least two singlets of the same player. It characterizes an opportunity for that player. "Diversity" counts the number of different singlet directions for each player.

Name	Description	dim M
	Feature sets for Nim-3	
$F0$	$f_i = p$, if i tokens left by player p, 0 else $(i = 0, \ldots, N-1)$	N
$F1$	$f_i = 1$, if i tokens left; $f_{i+N} = p$, if i tokens left $(i = 0, \ldots, N-1)$	$2N$
$F2$	$f_i = 1$, if i tokens left, 0 else; $f_N =$ player p $(i = 0, \ldots, N-1)$	$N+1$
	Feature sets for TicTacToe	
$T1$	$t_{0,1,2}$: number of singlets, doublets, triplets for $p = -1$; $t_{4,5,6}$: number of singlets, doublets, triplets for $p = +1$;	6
$T2$	$t_{0,1}$: number of singlets, doublets `X` if $p = -1$; 0 else $t_{2,3}$: number of singlets, doublets `O` if $p = +1$; 0 else $t_{4,5}$: diversity `O/X` if $p = -1$; 0 else $t_{6,7}$: diversity `O/X` if $p = +1$; 0 else $t_{8,9}$: crosspoint count `O/X`;	10
$T3$	same as $T2$ plus nine features containing the raw board position	19

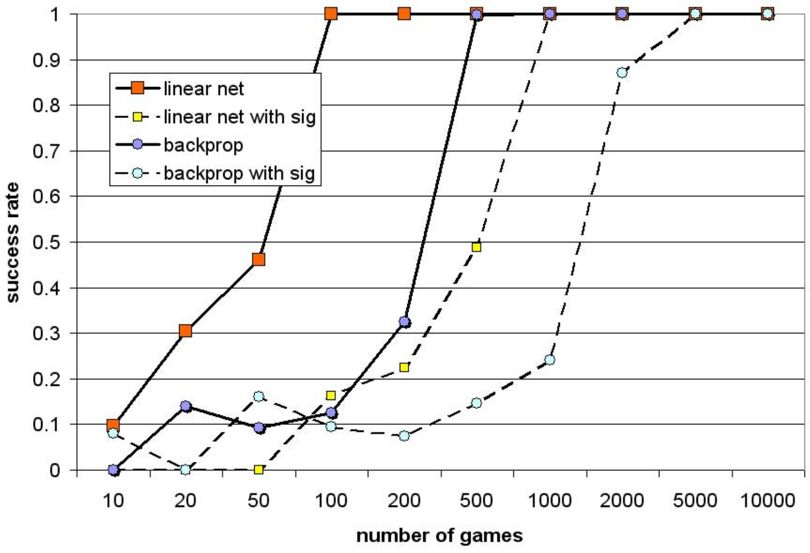


Fig. 2. This figure shows how fast different net types can learn Nim-3, as a function of the number of training games in self-play. The linear net without sigmoid in the output neuron learns ten times faster than the backprop net without sigmoid and 50 times faster than the backprop net with sigmoid. Parameters: $\alpha_{\text{init}} = 0.1$, $\alpha_{\text{final}} = 0.01$, $\lambda = 0$ and $\gamma = 0.9$. The backpropagation net has six sigmoidal hidden neurons.

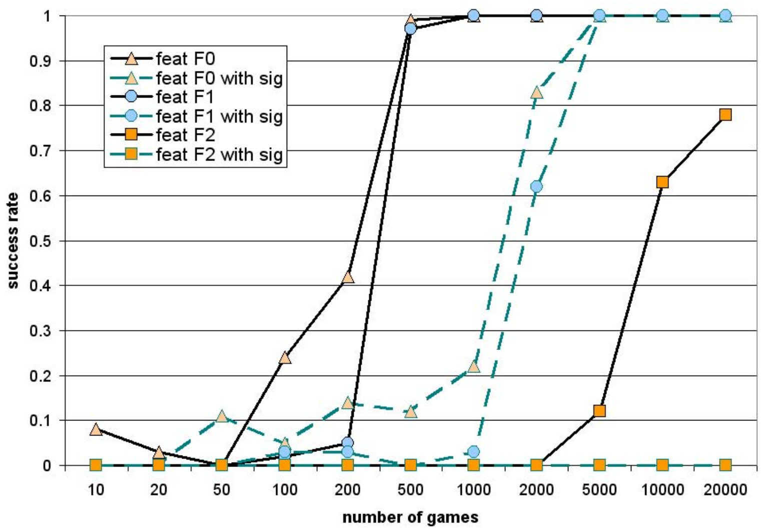


Fig. 3. Success rate in Nim-3 for different feature sets. Again nets without output sigmoid (solid lines) learn faster than those with (dashed lines). In all cases the net is a backpropagation net with six hidden neurons. The importance of correct feature-set selection is demonstrated: slow or no convergence on feature set $F2$. 12 hidden neurons produce similar results. Other parameters are the same as in Fig. 2.

Of course the linear net can not learn each feature-output-relation. While the feature set $F1$ is still linearly separable, the feature set $F2$ is not. The linear net can learn $F1$ as well, but not $F2$. But, as Fig. 3 shows from the average over 500 realisations, also the backpropagation net has increasing difficulties in learning $F1$ and $F2$. It does not succeed at all in the case $F2$, with sigmoid which is quite a surprising failure.

Figure 4 shows the second measurement metric, the percentage of correct moves on selected states. It is quite easy to reach 50% or more, but difficult to achieve a figure above 85% on the average of 200 independent realisations. *Single* realisations can reach 100% correct moves. The learning curve in Fig. 4 shows quite surprisingly a decline in performance for feature set $T1$ as G increases. As a general trend, the "richer" feature sets $T2$ and $T3$ show much better performance. The decline for $G \geq 10^4$ in 3 of 4 learning curves is not yet fully understood.

Finally we perform a TicTacToe tournament of 300 games between an RL agent, a minimax agent, and a random agent, where the latter chooses each move at random. The results shown in Tab. 2 are quite satisfactorily (no agent can do better than "tie" against the minimax agent). Although without any strategy, the random agent explores the full state space and it is not easy to win *consistently* against it. The percentages obtained here are about ten points higher (in the favor of RL) than the similar results reported in [4]. It has to be noted, that these results were achieved with a single (best) RL agent realisation, which is the same procedure as in [4].

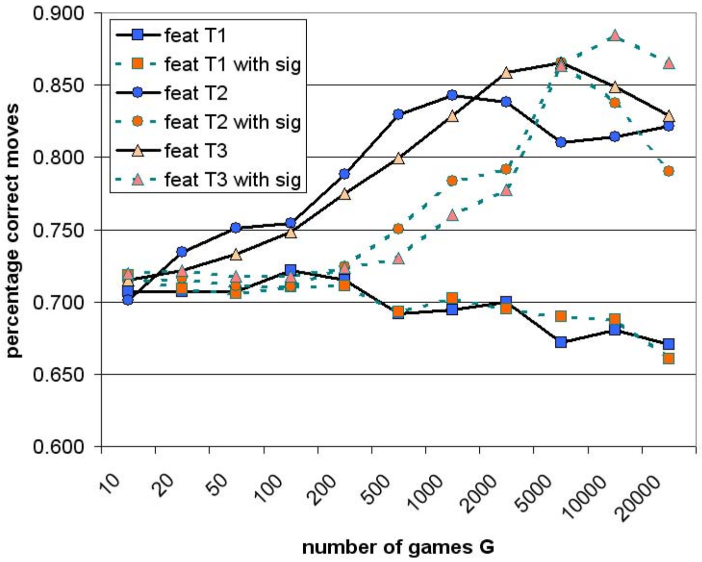


Fig. 4. Percentage of correct moves in TicTacToe for different feature sets. Each measurement point averages 200 independent realisations of a backpropagation net with 15 hidden neurons. Nets without output sigmoid (solid lines) learn faster in the initial phase, but nets with sigmoid (dashed lines) produce slightly better results as G increases. Best results are obtained with feature set $T3$. Other parameters are the same as in Fig. 2.

Table 2. Our results from a 300 games TicTacToe tournament. The RL agent is our backpropagation net with 15 hidden neurons, linear output function, trained on feature set $T3$ over $G = 10^4$ games of self-play. The minimax agent is a perfect player (recursive search of best move), while the random player chooses each move randomly. See Tab. 3 for comparable results by Stenmark [4].

X vs. O	X wins	tie	O wins
minimax vs. RL	0%	100%	0%
RL vs. minimax	0%	100%	0%
random vs. RL	0%	18%	82%
RL vs. random	100%	0%	0%

4 Discussion

An important result is that a linear output neuron is advantageous in nearly all cases compared to an output neuron with sigmoid function. This holds both for the linear net and the backpropagation net. It is surprising at first glance, since a sigmoidal output in $[0, 1]$ seems more appropriate for a function approximating $V(\boldsymbol{s})$, the probability of a win for player X. The reason for slower learning convergence (or no learning success at all) might lie in the following fact: In the

Table 3. Results as reported by Stenmark [4] on a TicTacToe tournament. The RL agent is a backpropagation net and was trained with 1 million games of self-play, yet it does not achieve the same performance as in Tab. 2. Entries in **bold face** highlight differences to Tab. 2.

X vs. O	X wins	tie	O wins
minimax vs. RL	0%	100%	0%
RL vs. minimax	0%	100%	0%
random vs. RL	**4.5**%	22%	**73.5**%
RL vs. random	**90.5**%	**9.5**%	0%

sigmoidal case the gradient $\nabla_{\boldsymbol{w}} f$ is a function proportional to $f \cdot (1-f)$ and thus becomes weaker as f approaches 0 or 1, the desired targets. Therefore "pulls" into the right direction have a smaller net effect than "pulls" into the wrong direction as they occur during the initial learning phase or while the correct concepts are not yet learned.

Another interesting result is that the right selection of features is of great importance to the learning process, as Fig. 4 shows. Too few features or features not specific enough towards the learning goal might block the road to success. On the other hand too many features *on top of specific features* seldom do any harm, the RL agent quickly learns to ignore irrelevant features. Even if an additional feature is quite unspecific (as for example the field contents used as extra features in set $T3$, which is on a single level not directly related to win or loss), it might help to make a formerly linearly inseparable task separable. This enables a linear function approximator to learn the desired behaviour quickly and robustly.

This brings in front another topic which is well-expressed on Suttons RL FAQ page [5], but too often forgotten in RL applications in the literature: Sutton emphasizes the robustness and speed of linear nets and prefers them in first approaches to new RL-tasks as opposed to backpropagation or other non-linear function approximators. We feel in the same way and think that the results presented here might assure these statements.

It pays off to think about features and their connection to the learning goal. The Nim-3 task and the seemingly similar feature sets $F0$ and $F2$ show that tiny modifications can be important: While $F0$ contains the same information as $F2$, it makes learning much easier. In the set $F2$ conflicting concepts are overlapping and hinder the learning process. Note that in the set $F0$ the input $f_4 = +1$ always signals a win for player $+1$, while in the set $F2$ the input $f_4 = 1$ means a win if $f_N = +1$, but a loss if $f_N = -1$. The net has to learn the "concept" $(f_4 \cdot f_N)$, but the conflicting TD error signals hinder it to do so. A similar source of conflicts, namely the interference of redundant inputs was reported by Togelius et al. in their work on memetic climbers [6].

Finally we compare our results with other work: For the game TicTacToe many RL-implementations exist [4,7]. The RL agent from [4] shows a 10% weaker performance on the random agent (Tab. 3), although it was trained 100 times

Table 4. Tournament results when playing TicTacToe against the agent ANN by Levkovich [7] which uses also RL and a feature set equivalent to our set $T1$. RL($T1$) and RL($T3$) are our RL agents with feature sets $T1$ and $T3$, resp. Other parameters are the same as in Tab. 2.

X vs. O	X wins	tie	O wins
ANN vs. RL($T3$)	7.4%	46%	46.6%
RL($T3$) vs. ANN	61%	18%	21%
ANN vs. RL($T1$)	24.1%	45.5%	30.4%
RL($T1$) vs. ANN	63.4%	16.3%	20.3%
RL($T1$) vs. RL($T3$)	16.2%	39.4%	44.4%

longer (1 million games). But the difference is that their input was only the set of raw board positions. This demonstrates the importance of feature inputs. The RL agent in [7] is available as source code, so we ran several direct tournaments where both agents had the same number $G = 10^4$ of training games (Tab. 4).[1] The win rate of our RL agent with feature set $T3$ was on average three or seven times higher than that of the RL agent ANN in [7], depending on whether our RL agent played as O or as X, resp. The performance of our RL agent with feature set $T1$ was a bit weaker, still slightly above ANN. So RL($T3$) leaves the tournament as the best agent, even stronger than RL($T1$).

As a general remark it is quite surprising that the different RL agents do not reach very often a tie or draw when playing against each other, as they do when playing against the perfect minimax agent or against themselves. The reason is probably that they did not encounter all variants of the other RL agent during self-play training, so both sides have their "vulnerabilities" when playing against each other. However, a better learning scheme seems theoretically possible where an agent learns a perfect strategy just from self-play. Yet it has not been achieved in an RL scheme *with* function approximation (where a learning step for state A can also influence the results for state B).

5 Conclusion and Future Work

Some insight has been gained in the way to configure RL learning agents. It has been studied in the case of strategic games but might as well be applicable to other control or learning problems with delayed rewards. A somewhat surprising result is that a sigmoidal output function is disadvantegeous in some tasks. Another interesting failure is the decline of the RL agent to learn the Nim-3 task from feature set $F2$, while rapidly converging on the very similar feature set $F1$. This shows the importance of the right feature selection. Compared to other RL solutions on the TicTacToe task we find good results within only 1/100 of the learning steps reported in [4].

[1] The source code of our implementation can be requested from the authors as well.

We plan to apply the results obtained here to more complex learning tasks, e.g., to the game Connect4 (state space complexity 10^{14}). A number of parameters and algorithmic choices have to be tuned carefully, which we plan to do in a systematic way with *Sequential Parameter Optimization* (SPO), a recent and leading technology in statistical analysis [8]. The most interesting "parameter" seems to be the design of a sufficiently rich and goal specific feature set for a learning task. It seems interesting to develop automatic or semi-automatic procedures for feature selection and test their validity on different RL learning tasks. Guidelines for the design of feature spaces could be the following properties:

– Is the feature distinctive with respect to the optimization goal, i.e., does at least one of the feature values reliably signal a win / a loss?
– Can we generate complex features as combinations of primitive features which have increased distinctiveness?
– Does a certain feature vector see too much spread in desired target values during learning? If so, probably different concepts of the learning task are mapped to the same feature vector and it might help to enrich the feature set to make these concepts separable.

It is desirable to find meta strategies for the selection of the best feature sets independent from the learning tasks. We plan to use again SPO [8] for this task. The final goal is to develop RL agents which learn optimal behaviour from the interaction with the environment in a way more robust and faster than the current RL agents.

References

1. Sutton, R.S.: Learning to predict by the method of temporal differences. Machine Learning 3, 9–44 (1988)
2. Tesauro, G.: TD-gammon, a self-teaching backgammon program, achieves master-level play. Neural Computation 6, 215–219 (1994)
3. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
4. Stenmark, M.: Synthesizing board evaluation functions for connect4 using machine learning techniques. Master's thesis, Østfold University College, Norway (2005)
5. Sutton, R.S.: Reinforcement learning FAQ (2008), Cited 20.4.2008, http://www.cs.ualberta.ca/sutton/RL-FAQ.html
6. Togelius, J., Gomez, F., Schmidhuber, J.: Learning what to ignore: Memetic climbing in weight and topology space. Congress on Evolutionary Computation (to appear, 2008)
7. Levkovich, C.: Temporal difference learning project (2008), Cited 10.3.2008, www.geocities.com/chen_levkovich/tdlearningproject.html
8. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation—The New Experimentalism. Natural Computing Series. Springer, Heidelberg (2006)

Evolvable Agents in Static and Dynamic Optimization Problems

J.L.J. Laredo[1], P.A. Castillo[1], A.M. Mora[1], J.J. Merelo[1], A. Rosa[2], and C. Fernandes[1,2]

[1] Department of Architecture and Computer Technology
University of Granada, Spain
{juanlu,pedro,amorag,jmerelo}@geneura.ugr.es
[2] LASEEB-ISR/IST. University of Lisbon, Portugal
{acrosa,cfernandes}@laseeb.org

Abstract. This paper investigates the behaviour of the Evolvable Agent model (EvAg) in static and dynamic environments. The EvAg is a spatially structured Genetic Algorithm (GA) designed to work on Peer-to-Peer (P2P) systems in which the population structure is a small-world graph built by newscast, a P2P protocol. Additionally to the profits in computing performance, EvAg maintains genetic diversity at the small-world relationships between individuals in a sort of social network. Experiments were conducted in order to assess how EvAg scales on deceptive and non-deceptive trap functions. In addition, the proposal was tested on dynamic environments. The results show that the EvAg scales and adapts better to dynamic environments than a standard GA and an improved version of the well-known Random Immigrants Genetic Algorithm.

1 Introduction

The natural evolution has shown to succeed in the changing conditions of the environments. Within any species, individuals' mating is spatially constrained, taking place between the fittest known individuals rather than just the fittest of the whole membership. This way, nature finds a way to preserve genetic diversity and responds to environmental changes (e.g. same species in different latitudes adapt to the climatic conditions).

Within the Evolutionary Algorithms (EAs) area, spatially structured EAs (ssEA) mimic these spacial relationships in nature (see e.g. [15] for a survey), but they remain still unexplored in the context of non-stationary environments. Neighbourhood structures in ssEA are modeled as a graph in which the vertices are individuals and edges represent relationships between them. The impact of different neighbourhood structures on the selection pressure has been studied for regular lattices [5] and different graph structures such as a toroid [6] or small-world [11]. The small-world structure has shown empirically to be competitive against panmictic EAs. Specifically in [7], a Watts-Strogatz structured population yields better results than a Barabási-Albert one and standard panmictic approaches.

Nevertheless, to the extent of our knowledge, there are just two works applying ssEAs to Dynamic Optimization Problems (DOP). The first one, proposed by Sarma and De Jong in [12], explored the behaviour of a cellular Genetic Algorithm (cGA) in a non-stationary environment. More recently, Alba et al. [2] did a most exhaustive investigation but then again over a cGA using a regular 2-dimensional lattice. Following this line, in this paper we analyse a small-world population applied to DOPs. Such a population structure is based on the newscast Peer-to-Peer (P2P) protocol presented in [8].

An additional advantage of using P2P protocols as population structure is that they are inherently designed to tackle large-scale graphs and they present consequently a good scalability behaviour. In order to assess the influence of such a population structure in an EA, we have performed a scalability study on static trap functions. Results show that our proposal scales better than a standard GA (a generational 1-elitism GA) which has been used as a baseline for comparison. Besides, results on DOP show that our approach also outperforms the standard GA and the state of the art algorithm Self-Organizing Random Immigrants GA (SORIGA) presented in [14]. SORIGA adopts a Self-Organized Criticality model in order to maintain a sub-population of random individuals and their offspring which varies in size by a power-law distribution.

The key to our P2P EA is the Evolvable Agent Model (EvAg) presented in [10]. It consists of a fine grained approach for parallelizing EAs in which there is a population of concurrent and self-scheduled agents performing the evolutionary steps of selection, variation and evaluation of individuals. Within such a study, several kinds of topologies were tested, concluding that a newscast based topology yields better results than topologies based on the Watts-Strogatz model or panmictic approaches.

The asynchronous update of the population in our approach implies a bias error during changes in DOPs (i.e. the change happens and some individuals might not be reevaluated during the first generation due to the asynchronous updating). In order to tackle with such an issue, we have assumed that the algorithm is able to detect changes. Additionally, we have also performed an exploratory study without change detections using a reduced test case in which the periods between changes are large (i.e. the larger the period is, the smaller the bias error).

The rest of the paper is structured as follow: Section 2 presents the algorithmic details of our proposal. In Section 3, we explore the performance of our approach in static and dynamic trap functions. Finally, in Section 4 we reach some conclusions and propose some future work lines.

2 Description of the Algorithm

The overall procedure of our approach consists of a population of Evolvable Agents (EvAg), described in Section 2.1, whose main design objective is to carry out the main steps of evolutionary computation: selection, variation and evaluation of individuals [4]. Each EvAg is a node within a neighbourhood in which the selection takes place locally. A decentralized system would be inefficient supporting a global comparison among all individuals. Consider, for example, roulette wheel or rank-based selection.

In this paper, we analyse the effects of a neighbourhood based on newscast, a self-organized small-world graph presented in [8].

2.1 Evolvable Agent

An *Evolvable Agent (EvAg)* itself is an EA composed of a single individual [9,10]. In spite of the model not having a *population* in the canonical sense, adjacent EvAgs provide each other with the genetic material that they require to evolve. Therefore, we talk about a population of EvAgs instead of a population of individuals.

Algorithm 1 shows the pseudo-code of an EvAg where the agent owns an evolving solution (S_t).

Algorithm 1. Evolvable Agent

```
$S_t \Leftarrow$ Initialize Agent
loop
  Sols $\Leftarrow$ Local Selection(Newscast) See algorithm 2
  $S_{t+1} \Leftarrow$ Recombination(Sols,$P_c$)
  $S_{t+1} \Leftarrow$ Evaluate($S_{t+1}$)
  if $S_{t+1}$ better than $S_t$ then
    $S_t \Leftarrow S_{t+1}$
  end if
end loop
```

The selection takes place locally into a given neighborhood where each agent select other agents' current solutions (S_t). Selected solutions are stored in $Sols$ ready to be recombined. Within this process a new solution S_{t+1} is generated. If the newly generated solution S_{t+1} is better than the old one S_t, it replaces the current solution.

2.2 Population Structure

In principle, our method places no restrictions in the choice of population structure, although this choice will have an impact on the dynamics of the algorithm since it establishes the environmental selection pressure. In this paper, we apply the newscast protocol as graph structure. Within this section we do not enter on the dynamics but on its functioning elements (see [8,16] for further details). Algorithm 2 shows the pseudo-code of the main tasks in the self-organized process which builds the newscast graph. Each node maintains a cache with one entry per node in the network at most. Each entry provides the following information about a foreign node: Time-stamp of the entry creation (it allows the replacement of old items) and an agent identifier.

Algorithm 2. Newscast protocol in node $EvAg_i$

```
Active Thread
while $EvAg_i$ not finished do
    sleep $\Delta T$
    $EvAg_j \Leftarrow$ Random selected node from $Cache_i$
    send $Cache_i$ to $EvAg_j$
    receive $Cache_j$ from $EvAg_j$
    $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end while

Passive Thread
while $EvAg_i$ not finished do
    wait $Cache_j$ from $EvAg_j$
    send $Cache_i$ to $EvAg_j$
    $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end while

Local Selection(Newscast)
$[EvAg_h, EvAg_k] \Leftarrow$ Random selected nodes from $Cache_i$
```

There are two different tasks that the algorithm carries out within each node. The active thread which initiates communications and the passive thread that waits for the answer. In addition, the local selection procedure provides the EvAg with other agents' current solutions ($EvAg_h(S_t)$ and $EvAg_k(S_t)$). After ΔT time each $EvAg_i$ initiates a communication process (active thread). It selects randomly an $EvAg_j$ from $Cache_i$ with uniform probability. Both $EvAg_i$ and $EvAg_j$ exchange their caches and merge them following an aggregation function. In our case, the aggregation consists of picking up the newest items (newscast) for each cache entry in $Cache_i$, $Cache_j$ and merging them into a single cache that $EvAg_i$ and $EvAg_j$ will share. We have fixed ΔT to once per evaluation.

The cache size plays an important role in the newscast algorithm. It represents the maximum number of connections (edges) that a node could have. For example, a topology with n nodes and a cache size of n, will lead to a complete graph topology. Therefore, the cache size is smaller than the number of nodes (typically around $log(n)$) in order to get small-world features such as a small charasteristic path length and a high clustering coefficient (for further details on the dynamics refer to [16]). We have fixed the cache size to 4 within the experimental setup.

3 Experimental Setup and Results

In order to investigate how EvAg behaves on static and dynamic environments experiments were conducted on trap functions [1]. A trap function is a piecewise-linear function defined on unitation (the number of ones in a binary string). There are two distinct regions in search space, one leading to a global optimum and the other leading to the local optimum. In general, a trap function is defined by the following equation:

$$trap(u(\overrightarrow{x})) = \begin{cases} \frac{a}{z}(z - u(\overrightarrow{x})), if \quad u(\overrightarrow{x}) \leq z \\ \\ \frac{l}{l-z}(u(\overrightarrow{x}) - z), \quad otherwise \end{cases} \quad (1)$$

where $u(\overrightarrow{x})$ is the unitation function, a is the local optimum, b is the global optimum, l is the problem size and z is a slope-change location separating the attraction basin of the two optima.

For the following experiments, 2-trap, 3-trap and 4-trap functions were designed with the following parameter values: a = l-1; b = l; z = l-1. With these settings, 2-trap is not deceptive, 4-trap is deceptive and 3-trap lies in the region between deception and non-deception. Under these conditions, it is possible not only to examine how EvAg scales on trap functions, but also to investigate how the scalability varies when changing from non-deceptive to deceptive search landscapes. Scalability tests were performed by juxtaposing m trap functions and summing the fitness of each sub-function to obtain the total fitness. For each trap and each size m the bisection [13] method was used to determine the optimal EvAg population size N (the lowest N for which 98% of the runs solve all the traps). As stated in the bisection method, mutation rate was set to 0, the idea is to calibrate a minimum population size such that using random initialization it is able to provide enough building blocks to converge to the optimum without other mechanism than recombination. EvAg was tested with $p_c = 1.0$, uniform crossover and binary tournament. With respect to DOPs, population size N was set to 240. In order to evaluate EvAg's results a standard generational GA (GGA) with 1-elitism was also tested with the same parameter values. Results are depicted in figure 1.

From the graphics in figure 1 it can be concluded that EvAg scales better than GGA on 2, 3 and 4-trap, but the improvement is much more noticeable when solving the deceptive 4-trap function. Under these conditions (4-trap), a GGA faces extreme difficulties because lower order building blocks mislead the search towards local optima instead of combining to form higher order building-blocks, thus challenging the GA's search mechanisms, and if the problem size grows, the computational effort grows exponentially. A possible explanation for EvAg's better scalability lies in its ability to maintain genetic diversity at a higher and consequent reduction of its optimal population size (N). With a lower optimal N, EvAg needs fewer evaluations to reach the optimum, when compared to standard GAs. Investigating scalability is of extreme importance when changing from a "toy problem" test environment to real-world problems which may require very large chromosomes to codify the solutions.

Experiments on dynamic problems were also conducted with trap functions. For that purpose, the DOP generator presented in [17] was used to build different changing environments based on 3- trap and 4-trap functions. Given a stationary problem $(f(x)(x \in \{0, 1\}^L))$ where L is the chromosome length, DOPs may be

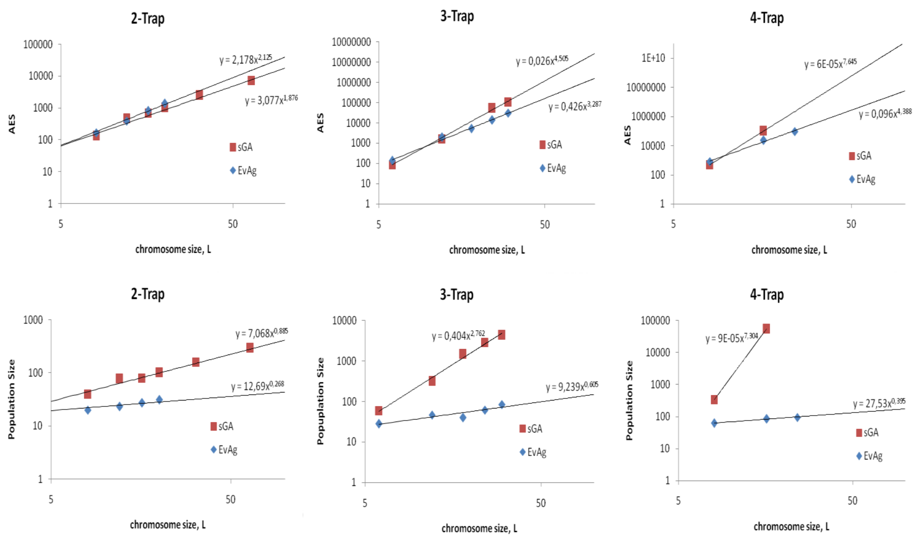


Fig. 1. Scalability with trap functions. Optimal population size and Average Evaluations to Solution (AES) values for a standard GA (sGA) and the Evolvable Agent Model (EvAg).

designed by applying a binary mask to each solution before its evaluation in the following manner:

$$f(x,t) = f(x \quad XOR \quad M(k)) \tag{2}$$

where t is the generation index, k = tτ is the period index and f(x, t) is the fitness of the string x. M(k) is incrementally generated as follows:

$$M(k) = M(k-1) \quad XOR \quad T(k) \tag{3}$$

where $T(k)$ is an intermediate binary mask for every period k. T(k) has $\rho\times$L ones. ρ is a value between 0 and 1 that controls the intensity, or severity, of changes (i.e. $\rho = 0$ stands for a stationary problem and $\rho = 1$ represents the highest degree of change). Therefore, by setting ρ and τ it is possible to control two of the most important features of DOPs test environments: severity (ρ) and speed (τ) of change [3]. Nine different scenarios for each trap were designed by setting ρ to 0.05, 0.6 and 0.95, and τ to 10, 100 and 200 generations. Stationary functions were designed with 10 subfunctions each, meaning that size of dynamic 3-trap is L = 30 and size of dynamic 4-trap is L = 40. EvAg, GGA and SORIGA were tested with uniform crossover, bit-flip mutation, binary tournament, $p_c = 1.0$, $N = 240$ and 1-elitism (GGA). SORIGA's parameter r_r was set to 3.

GAs performance analysis on DOPs must be addressed in a different manner from static environments' usual procedure. Dynamic behaviour throughout the run must be examined, rather than the final convergence. For that purpose, the

Table 1. Results on dynamic 3-trap and 4-trap (averaged over 30 independent runs). Mean of *best of generation* and corresponding standard deviation values.

	τ	10			100			200		
	ρ	0.05	0.6	0.95	0.05	0.6	0.95	0.05	0.6	0.95
3-trap	GGA	25.72	21.88	24.19	29.4	26	25.57	29.81	26.7	25.6
	$(p_m = \frac{1}{L})$	±0.97	±0.27	±0.24	±0.44	±0.33	±0.17	±0.11	±0.33	±0.22
$L = 30$	SORIGA	**26.43**	22.74	24.10	**29.74**	**27.71**	26.67	**29.86**	**28.8**	27.95
	$(p_m = \frac{1}{2L})$	±0.62	±0.3	±0.26	±0.05	±0.17	±0.16	±0.02	±0.1	±0.16
	EvAg	25.71	**22.83**	**26.37**	28.91	27.04	**27.83**	29.61	27.64	**28.03**
	$(p_m = \frac{1}{L})$	±0.38	±0.43	±0.34	±0.26	±0.17	±0.39	±0.27	±0.2	±0.33
4-trap	GGA	28.92	26.63	31.66	30.52	32.55	35.1	30.61	33.08	35.3
	$(p_m = \frac{1}{L})$	±0.33	±0.39	±0.52	±0.57	±0.34	±0.11	±0.44	±0.29	±0.129
$L = 40$	SORIGA	28.64	26.54	29.4	34.12	32.4	34.7	**35.92**	33.43	35.02
	$(p_m = \frac{1}{8L})$	±0.71	±0.36	±0.55	±1.57	±0.37	±0.14	±1.4	±0.38	±0.09
	EvAg	**31.71**	**27.82**	**32.71**	**34.89**	**33.76**	**36.9**	35.32	**34.87**	**37.17**
	$(p_m = \frac{1}{L})$	±0.53	±0.39	±0.46	±0.5	±0.28	±0.31	±0.51	±0.25	±0.33

evaluation of the algorithmic performance is done by measuring the mean best-of-generation values (this is the standard procedure for DOPs). In addition, the progression of best-of-generation values may be plotted in a graph, thus helping to understand how the algorithm reacts to changes in the environment. Different mutations rates were tested, and results in table 1 show the best configurations, that is, the mutation rates that attained the higher values when averaging the mean best-of-generation of the nine scenarios.

Table 2 helps to understand the relevance of the results in table 1 by showing the results of pairwise t-test that compares the algorithms' performance. The (+) sign means that algorithm 1 is significantly better than algorithm 2, ($\sim$) means that the performance is equivalent and ($-$) means that the second GA is better. While in 3-traps GGA and SORIGA still outperform EvAg in some scenarios, in 4-traps our proposal achieves better results, with statistical significance, in all the scenarios except one. EvAg abilities to solve DOPs appear to emerge when facing a harder problem for GAs.

Figure 2 shows the dynamic behaviour of EvAg and GGA throughout the run. It is clear that EvAg is more able to track the optimum, maintain a lower distance to the solution during the search, in all scenarios. When $\rho = 0.95$, GGA oscillates between local and global optimum, without really tracking the solution, while EvAg maintains the best fitness closer to the global optimum, which is $f(x) = 40$.

All this results have been obtained assuming that the algorithm is able to detect changes in DOPs. However, some real situations would not allow our proposal to detect changes. Therefore, looking forward future improvements in the model, we have analysed it without change detections in the 4-trap function using the larger periods between changes ($\tau = 100$ and $\tau = 200$). Note that the results in Tables 3 and 4 are given with a bias error of 0.008% and 0.004% respectively for $\tau = 100$ and $\tau = 200$. Besides, the fact of changes are not detectable means that EvAg will need extra computational effort. For that reason it is not surprising that EvAg performs better if we assume change detections (see table 4). But when compared to GGA and SORIGA, EvAg still performs

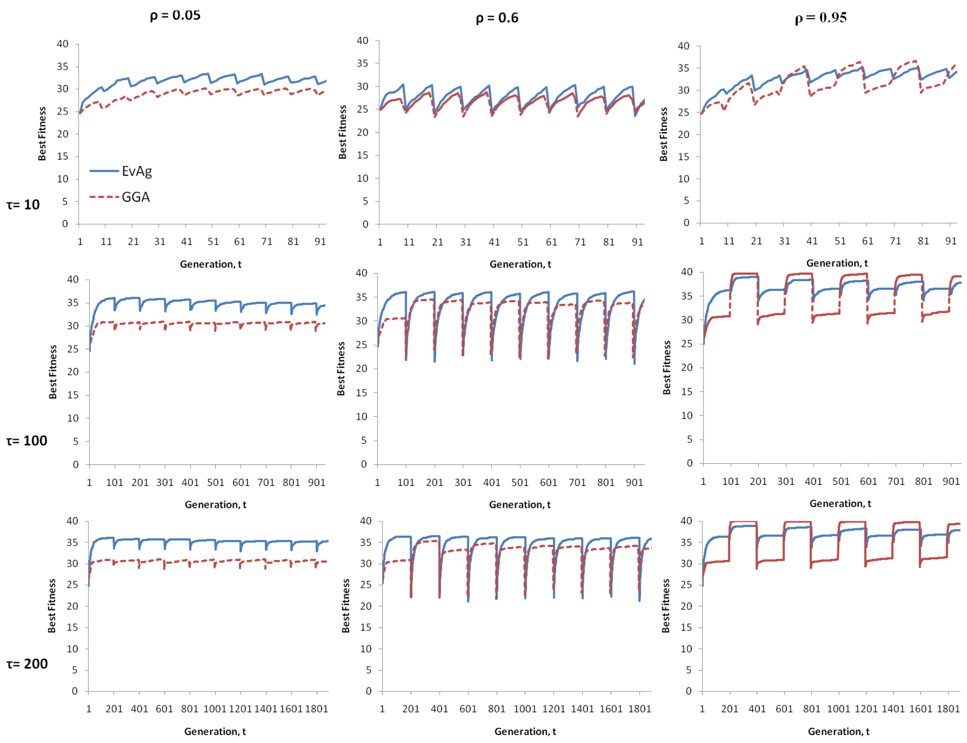


Fig. 2. Dynamics when tracking 4-trap functions ($L = 40$). *Best of generation* curves

Table 2. Pairwise *t-test* on dynamic 3-trap and 4-trap. Evolvable Agent vs. GGA and SORIGA.

t-test	τ	10			100			200		
	ρ	0.05	0.6	0.95	0.05	0.6	0.95	0.05	0.6	0.95
3-trap	EvAg vs. GGA	$\sim$	+	+	−	+	+	−	+	+
	EvAg vs. SORIGA	−	$\sim$	+	−	−	+	−	−	+
4-trap	EvAg vs. GGA	+	+	+	+	+	+	+	+	+
	EvAg vs. SORIGA	+	+	+	+	+	+	+−	+	+

Table 3. Results on dynamic 4-trap. It includes Evolvable Agent without change detections. Mean of *best of generation* and corresponding standard deviation values.

	τ	100			200		
	ρ	0.05	0.6	0.95	0.05	0.6	0.95
4-trap	Without changes detection	34.6	32.6	36.6	34.9	34	37
	($p_m = \frac{1}{L}$)	±0.4	± 0.38	±0.3	±0.47	±0.27	± 0.23
$L = 40$	Changes detection	**34.89**	**33.76**	**36.9**	35.32	**34.87**	**37.17**
	($p_m = \frac{1}{L}$)	±0.5	±0.28	±0.31	±0.51	±0.25	±0.33
	GGA	30.52	32.55	35.1	30.61	33.08	35.3
	($p_m = \frac{1}{L}$)	±0.57	±0.34	±0.11	±0.44	±0.29	±0.129
	SORIGA	34.12	32.4	34.7	**35.92**	33.43	35.02
	($p_m = \frac{1}{8L}$)	±1.57	±0.37	±0.14	±1.4	±0.38	±0.09

Table 4. Pairwise *t-test* on dynamic 4-trap. Evolvable Agent without change detections vs. GGA, SORIGA and EvAg assuming that changes are detectable.

t-test 4-trap	τ	100			200		
EvAg without change detections	ρ	0.05	0.6	0.95	0.05	0.6	0.95
vs. EvAg change detection		−	−	−	−	−	−
vs. GGA		+	∼	+	+	+	+
vs. SORIGA		+	+	+	+−	+	+

better (if changes are not detectable, all the agents' solutions must be evaluated in each generation, even if the fitness has been already computed).

4 Conclusions and Future Works

In this paper we have investigated the scalability of the Evolvable Agent Model in static trap functions and its ability for responding to changes in dynamic optimization problems. Results show that our approach scales better than a standard GA and is able to outperform SORIGA [14], one of the state of the art algorithms in DOPs. These results are specially remarkable under deceptive conditions. The key to this is a population structure based on the small-world graph built from a P2P protocol.

The non-generational procedure of the EvAg model produces a bias error during changes in DOPs that we have avoid by assuming that the algorithm is able to detect changes. Additionally, we have also explored its behaviour without change detections. To that end, we have used the test cases in which the periods between changes are large and the bias error is minimum. In both cases our approach outperforms the standard GA and SORIGA.

As a future work, we intend to find a mechanism to avoid such an error without assuming that changes are detectable. It will allow us to explore a wider range of more realistic non-stationary problems.

Acknowledgements

This work has been supported by the Spanish MICYT project TIN2007-68083-C02-01, the Junta de Andalucia CICE project P06-TIC-02025 and the Granada University PIUGR 9/11/06 project.

References

1. Ackley, D.H.: A connectionist machine for genetic hillclimbing. Kluwer Academic Publishers, Norwell (1987)
2. Alba, E., Badia, J.S., Luque, G.: A Study of Canonical GAs for NSOPs. In: Meteheuristics. Operations Research/Computer Science Interfaces, vol. 39, pp. 245–260. Springer, US (2007)

3. Angeline, P.J.: Tracking extrema in dynamic environments. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213. Springer, Heidelberg (1997)
4. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
5. Giacobini, M., Tomassini, M., Tettamanzi, A., Alba, E.: Selection intensity in cellular evolutionary algorithms for regular lattices. IEEE Transactions on Evolutionary Computation 9(5), 489–505 (2005)
6. Giacobini, M., Alba, E., Tettamanzi, A., Tomassini, M.: Modeling selection intensity for toroidal cellular evolutionary algorithms. In: EWSPT 1996. LNCS, vol. 1149, pp. 1138–1149. Springer, Heidelberg (1996)
7. Giacobini, M., Preuss, M., Tomassini, M.: Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In: Gottlieb, J., Raidl, G.R. (eds.) EvoCOP 2006. LNCS, vol. 3906, pp. 85–96. Springer, Heidelberg (2006)
8. Jelasity, M., van Steen, M.: Large-scale newscast computing on the Internet. Technical Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands (October 2002)
9. Laredo, J.L.J., Eiben, E.A., Schoenauer, M., Castillo, P.A., Mora, A.M., Merelo, J.J.: Exploring selection mechanisms for an agent-based distributed evolutionary algorithm. In: GECCO 2007, pp. 2801–2808. ACM Press, New York (2007)
10. Laredo, J.L.J., Castillo, P.A., Mora, A.M., Merelo, J.J.: Exploring population structures for locally concurrent and massively parallel evolutionary algorithms. In: IEEE Congress on Evolutionary Computation (CEC2008), WCCI 2008 Proceedings, June 2008, pp. 2610–2617. IEEE Computer Society Press, Los Alamitos (2008)
11. Preuss, M., Lasarczyk, C.: On the importance of information speed in structured populations. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 91–100. Springer, Heidelberg (2004)
12. Sarma, J., De Jong, K.A.: The behavior of spatially distributed evolutionary algorithms in non-stationary environments. In: Banzhaf, W., Daida, J.M., Eiben, E.A., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) GECCO 1999, Orlando, FL, USA, pp. 572–578. Morgan Kaufmann, San Francisco (1999)
13. Sastry, K.: Evaluation-relaxation schemes for genetic and evolutionary algorithms. Technical Report 2002004, University of Illinois at Urbana-Champaign, Urbana, IL (2001)
14. Tinós, R., Yang, S.: A self-organizing random immigrants genetic algorithm for dynamic optimization problems. Genetic Programming and Evolvable Machines 8(3), 255–286 (2007)
15. Tomassini, M.: Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time. Natural Computing Series. Springer, New York (2005)
16. Voulgaris, S., Jelasity, M., van Steen, M.: A Robust and Scalable Peer-to-Peer Gossiping Protocol. In: Moro, G., Sartori, C., Singh, M.P. (eds.) AP2PC 2003. LNCS (LNAI), vol. 2872, pp. 47–58. Springer, Heidelberg (2004)
17. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)

The Impact of Global Structure on Search

Monte Lunacek, Darrell Whitley, and Andrew Sutton

Colorado State University
Fort Collins, Colorado 80523, USA

Abstract. Population-based methods are often considered superior on multi-modal functions because they tend to explore more of the fitness landscape before they converge. We show that the effectiveness of this strategy is highly dependent on a function's *global structure*. When the local optima are not structured in a predictable way, exploration can misguide search into sub-optimal regions. Limiting exploration can result in a better non-intuitive global search strategy.

Keywords: Funnel landscapes, test functions, exploration, dynamic populations.

Many artificial test functions have a "big valley" topology, where a decrease in fitness implies that, on average, search is getting closer to the global optimum. Although the search space is highly multi-modal, the local optima are structured such that there exists a global trend toward the best solution. Problems that exhibit this characteristic are sometimes referred to as *single-funnel* landscapes.

There are several real-world applications that do not have this simple structure. Wales [7] suggests that many optimization problems in computational biology are difficult because local optima often form in distinct, spatially separate clusters within the search space. Problems of this type have multiple funnels, resulting in a landscape that has a less predictable underlying global structure.

The way that global structure impacts evolutionary search is not well understood, in part, because many of the test functions used for evaluation have single-funnel landscapes. There are also a few test functions that have multiple funnels, but the number of funnels increases with dimensionality. This complexity makes it difficult to understand search behavior in high dimensions.

We have several objectives in this paper. First, we describe a method for creating landscapes that contain exactly two funnels, regardless of the problem size. Then, we empirically show that several *evolution algorithms* have an extremely low probability of success when the global optima is located in a proportionally smaller funnel. Finally, we demonstrate that limiting exploration can result in a performance gain.

1 Motivation and Background

The degree to which an algorithm will perform well on an application partly depends on how well the algorithm can deal with the features that make the problem difficult. Researchers within the computational chemistry community have started to pay attention to how global structure affects problem difficulty [4]. Much of their attention has been devoted to studying Lennard-Jones clusters, which are a class of configuration optimization problems where the goal is to find the spatial positions for a set of atoms that has the smallest potential energy.

The energy surface of the Lennard-Jones potential is highly multimodal, and the most difficult instances have a *double-funnel* landscape. Assuming that a search algorithm can escape local optima, the underlying global structure of a problem may have a greater impact on problem difficulty than the number of local optima [6].

The Rastrigin function is a classic single-funnel landscape. Kern *et al.* [3] point out that there are two potential strategies for solving this highly multimodal problem. The first is to exploit *separability*, which reduces its difficulty to N one-dimensional lines searches, where N is the number of parameters. The other strategy is to exploit the problems global structure; CMA-ES [2] and Basin-Hopping [8] avoid local optima by exploiting an underlying structure. The main question we are exploring in this paper is: how does this underlying structure impact evolutionary search?

2 Creating Double-Funnel Landscapes

The relative merit in any empirical study is limited by how well we understand the characteristics that make realistic parameter optimization problems difficult, and by our ability to embed these features into benchmark test functions. In this section, we describe two *double-funnel* test problems. First, we create a simple surface comprised of *only* two quadratic spheres. Then, we take this simple surface and add local optima to it. This creates a multi-funnel surface similar to Rastrigin's function.

2.1 The Double-Sphere

The landscape structure of our simple *double-sphere* test function is the minimum of two quadratic functions, where each sphere creates a single funnel in the search space. The placement of each sphere is critical because the barrier that divides them will be inconsequential if they are too close. We also want this barrier height to scale with dimensionality.

To address these concerns, we place each quadratic sphere along the positive diagonal of the search space, which is bounded on the interval $[-5, 5]^N$. The optimal sphere is located in the middle of the positive quadrant of the search space, at $\mu_1 = 2.5$ in each dimension. The sub-optimal sphere is centered at $\mu_2 = -2.5$ across all dimensions. The distance between each funnel increases proportionally with dimensionality, and this construction creates an underlying surface that is *globally* non-separable.

Lennard-Jones double-funnel problems are difficult when 1) the sub-optimal funnels is nearly as deep as the optima funnel, and 2) the basin of attraction to the optimal funnel is small. We simulate this by increasing the height of the sub-optimal funnel by a value of d. That way, the value of the optimal funnel is unchanged. In order to change the relative size of each funnel, we scaled the sub-optimal funnel by a constant factor, denoted s. This way, the optimal funnel retains it shape regardless of scaling, and therefore, has a more consistent level of difficulty. Multiplying the sub-optimal funnel by a number greater than one will create a more narrow sub-optimal funnel. The

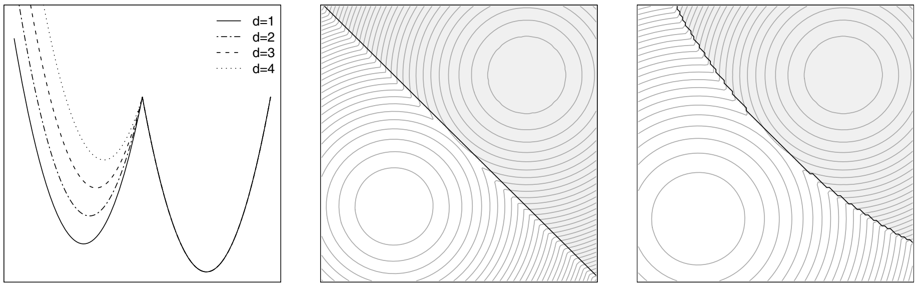


Fig. 1. The impact of d and s on the *double-sphere* function. Increasing d creates more distinction between the funnels (left). When $s = 0$ (middle), the two funnels are the same size. Decreasing s creates a larger sub-optimal funnel (right).

opposite is true when s is less than one. The overall form of our multi-funnel sphere function is:

$$f_{\text{double-sphere}}(\boldsymbol{x}) = \min\left(\sum_{i=1}^{N}(x_i - \mu_1)^2, \quad d \cdot N + s \cdot \sum_{i=1}^{N}(x_i - \mu_2)^2\right)$$

In order to make s the primary control characteristic for the size of each basin of attraction, we shifted the mean of the sub-optimal sphere such that the barrier between them, which is the point at which they intersect, is always located at the origin of the search space. This configuration requires $\mu_2 = -\sqrt{(\mu_1^2 - d)/s}$.

Values s and d control the size and depth of the sub-optimal funnel. The leftmost graph in Figure 1 is a diagonal slice showing how the different values of d impact the depth of the sub-optimal funnel. The middle and right-most contour plots illustrate the impact of s. The two funnels are the same size in the middle graph (e.g. $s = 1.0$), but the right-most graph creates a larger sub-optimal funnel (white) using $s = 0.7$. We use the quadratic penalty term described by Hansen and Kern [2] to enforce strong boundaries.

2.2 The Double-Rastrigin

We wanted a double-funnel test problem with properties similar to Rastrigin's function because it would isolate global structure as the main difference impacting problem difficulty on a problem that is well-understood. We create a double-funnel version of Rastrigin's function by adding local optima to the *double-sphere* function. We translate the cosine term used in Rastrigin's function by μ_1 so that the minimum of the local optima component is centered at the bottom of the optimal funnel. The overall form of the *double-Rastrigin* function is

$$f_{\text{double-Rastrigin}}(\boldsymbol{x}) = f_{\text{double-Sphere}}(\boldsymbol{x}) + 10\sum_{i=1}^{N}(1 - \cos 2\pi(x_i - \mu_1))$$

3 Understanding the Impact of Global Structure

In this section, we explore how the characteristics of the *double-sphere*, which we measure in terms of s and d, impact search. We compare a simple evolution strategy using *Cumulative Step-length Adaptation* (CSA-ES) [5,3], the *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES)[2], and the CHC genetic algorithm [1]. Please see citations for descriptions and parameter settings.

We measure performance in terms of *success rate*, which we denote as ω, and define as the probability that an algorithm will converge to the global optimum. In each experiment, we estimate ω by running 1000 trials of each algorithm and counting the number of instances that find the global optimum.

Our results show that population-based methods are vulnerable to the size of each funnel, as controlled by s, when the depth of the two funnels are relatively close. That is, there exists some funnel characteristics where the exploration process will misguide search into the biggest funnel, not the deepest.

This section is organized in the following way. First, we measure the performance of local search in order to get a rough estimate of the size of each basin of attraction over a range of s and d values. Then we investigate how CMA-ES and CHC perform on the double-sphere. Although this problem only has two local optima, we still find both algorithms can fail even when the size of the optimal basin of attraction is fairly large. Finally, we discuss why this is important from a global optimization perspective by evaluating CMA-ES and CSA-ES on the double-Rastrigin function.

3.1 Local Search Properties of the Double-Sphere

As a baseline, we use the success rate of a local search method, where the probability of finding the global solution is proportional to the size of the basin of attraction to the optimum. We start by considering the double-sphere with dimension $N = 30$. We vary s between $[0.2, 1.4]$ by increments of 0.1, and evaluate different sub-optimal depths of $d = 1, 2,$ and 3.

When we estimate $\hat{\omega}$ for local search, we find a positive and approximately linear relationship between $\hat{\omega}_{\mathrm{LS}}$ and s. That is, as we decrease s, we also decrease the probability of finding the global optimum using local search. This makes sense because a small s value increases the size of the sub-optimal funnel, making the optimal funnel proportionally smaller (e.g. the basin of attraction to the optimum is smaller). The left graph in Figure 2 shows the relationship between $\hat{\omega}_{\mathrm{LS}}$ and s. Notice that when $s = 1$, each funnel occupies $\approx 50\%$ of the search space (black dot).

We use this estimate of the size of each basin of attraction as a baseline for interpreting our results. That is, instead of graphing $\hat{\omega}$ for each algorithm as a function of s, we plot the $\hat{\omega}$ values as a function of $\hat{\omega}_{\mathrm{LS}}$, the estimate size of basin of attraction to the global optimum. This makes it easier to observe when the evolutionary search is under- or over-performing with respect to what we would expect from local search.

For example, Figure 2 also shows the success rates of CMA-ES using the default population size of $\lambda = 14$ (for $N = 30$). Since CMA-ES is always above the gray line, we can observe that the success rates for CMA-ES are greater than that of local search. However, there is still a strong *linear* relationship between $\hat{\omega}$ and the size of the optimal funnel.

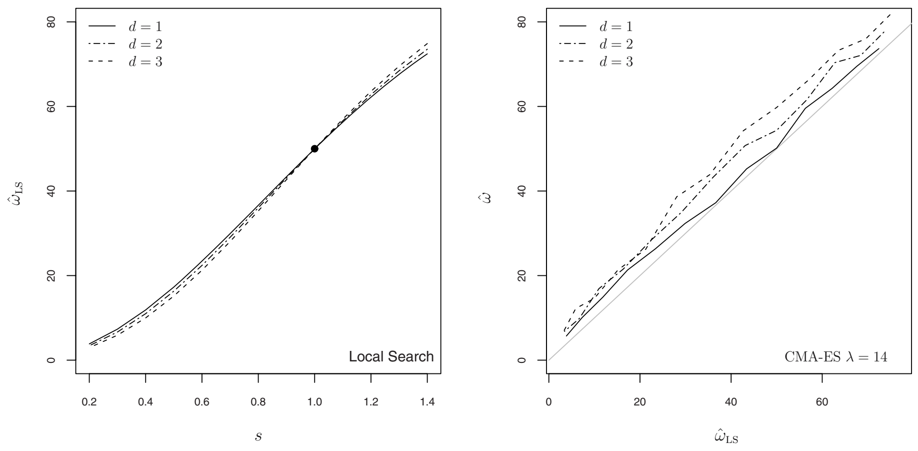


Fig. 2. Local search on the double-sphere: There is an approximately linear relationship between the size of the optimal funnel and success rate of local search, $\hat{\omega}_{\mathrm{LS}}$ (left). Notice that the depth of the sub-optimal funnel does not greatly impact $\hat{\omega}_{\mathrm{LS}}$. The right plot shows success rate for CMA-ES using the default population size. The success rates for CMA-ES are greater than that of local search, but still strongly tied to the size of the optimal funnel ($\approx \hat{\omega}_{\mathrm{LS}}$).

3.2 Global Search Properties of the Double-Sphere

Most evolutionary algorithms perform better on multimodal surfaces when they use a larger population size. This is especially true for CMA-ES [2,3] and CSA-ES [3]. CHC is probably the exception, as it was designed to use smaller populations [9].

In this section, we would like to understand how the double-sphere impacts global search. In the next section we will consider a range of population sizes for CSA-ES and CMA-ES, but for now, we fix the population size of CMA-ES to $\lambda = 500$. For CHC we use the default of population size of 50 with 10-bits of precision. Our results did not change dramatically with increased population sizes for CMA-ES or CHC. Changing the precision on CHC to $20-$bits also had little impact. A maximum of $100,000$ evaluations were allocated and *no random restarts* were used (expect for the soft-restarts used by CHC). We discuss the role of restarts in the next section.

We observe a similar probability distribution for each algorithm. Instead of a linear trend, as observed for the local search methods, the distribution is pulled into a sigmoid. When the optimal funnel is proportionally larger than the sub-optimal funnel, success rates are extremely high. However, when the optimal funnel is proportionally smaller, the success rates for CMA-ES and CHC drop dramatically. Figure 3 show the probability of success for CMA-ES and CHC as a function of the basin of attraction size.

Consider the extreme cases. When the relative size of the optimal funnel is $\approx 70\%$, evolutionary search is highly successful ($\hat{\omega} \approx 100\%$). This means that when local search finds the optimal solution $\approx 70\%$ of the time, CMA-ES and CHC will almost always find the optimal solution. On the other hand, when the relative size of the optimal

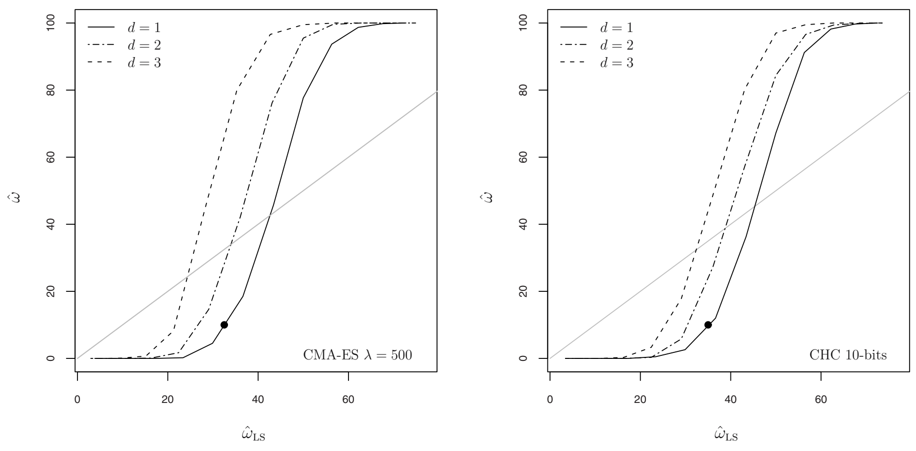


Fig. 3. CMA-ES and CHC on the double-sphere: The probability of success as a function of the size of the basin of attraction to the optimal funnel, as estimated with local search ($\hat{\omega}_{\text{LS}}$). The gray line indicates the success probability of local search. For each algorithm, the trend is similar; when the optimal funnel is relatively large, the success rates for evolutionary algorithms are high. When the relative size of the optimal funnel is low, evolutionary search is more likely to fail.

funnel is only $\approx 10\%$, CMA-ES and CHC fail to find the global optimum. This is true for all the d values we considered.

As we increase d, we increase the height of the sub-optimal funnel. Figure 3 show that larger values of d shift the $\hat{\omega}$ distribution to the left, meaning that a smaller s value, and therefore, a smaller basin of attraction to the global optimal, is required to observe failure. The hardest problems for CHC and CMA-ES are those where the depths of the two funnels are close ($d = 1$) and the basin of attraction to the optimal funnel is comparatively small (s is small).

The black dots in Figure 3 represent a success rate of 10% for each algorithm when $d = 1$. This means that CMA-ES will succeed less than 10% of the time even when the relative size of the optimal funnel is $\approx 33\%$. A similar problem occurs with CHC. Even when the basin of attraction to the global optima is $\approx 35\%$, the success rate for CHC is about 10%. In general, when local search finds the optimal solution $\approx 1/3$ of the time, the evolutionary algorithms we tested are likely to fail when the depths of the two funnels are relatively close.

The key observation we make in this section is this: when the depths of two funnels are close (e.g. $d = 1$, about 17% different from the barrier that divides them), the global search parameter settings employed by the evolutionary algorithms we tested are likely to cause failure, even when the optimal basin of attraction is relatively large, $\approx 30\%$. As we increase the depth of the sub-optimal funnel, evolutionary search is more successful.

3.3 Implications for Global Search: Double-Rastrigin

Considering that CMA-ES using the default population size has probabilities of success that are similar to, or even better than, that of local search, why should we care about the

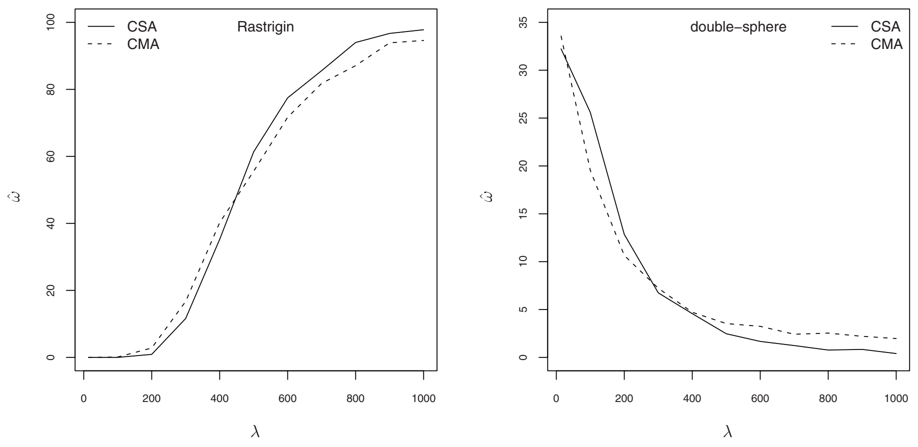


Fig. 4. Increasing the population size (λ) increases the probability that each evolution strategy will find the optimal solution on Rastrigin's' function (left), but decreases the probability of success on the double-Sphere function

bias of larger populations? The main reason this matters is that if an algorithm cannot cope with the simple structure of the double-sphere, it will also not be successful on more complex multimodal surfaces, like the double-Rastrigin, where the double-sphere dictates the underlying global structure.

We consider three 30-dimensional functions: Rastrigin, double-sphere, and double-Rastrigin. For the double-sphere and the double-Rastrigin, we created instances that are intentionally difficult for CMA-ES by choosing $d = 1$ and $s = 0.7$, which corresponds to an $\hat{\omega}_{\text{LS}} \approx 30\%$. We only consider CSA-ES and CMA-ES because they have strong termination criteria and can solve the 30-dimensional Rastrigin function with large populations. This simplifies the interpretation of our results.

The leftmost graph in Figure 4 shows the estimated success rates for the ES algorithms, *without restarts*, on Rastrigin's function as the population varies from $[100, 1000]$ by increments of 100. We have also included the default population size of $\lambda = 14$. These results are consistent with previously reported success rates [2,3]. The noticeable trend is that larger populations are more able to exploit the underlying sphere structure of the Rastrigin function and locate the best solution. Smaller population sizes tend to get stuck in one of the many local optima. For example, CMA-ES with a population of $\lambda = 14$ never finds the global solution. Using a population size of $\lambda = 100$, CMA-ES only finds the optimal about 10 out of 1000 times.

As we vary the population size for the ES algorithms on the double-sphere, we find the opposite is true. High success rates are realized with low population sizes, but larger values of λ cause CSA-ES and CMA-ES to exhibit extremely low success rates. The right-most graph in Figure 4 shows these results.

This presents an interesting trade-off for the double-Rastrigin function: find a population size that balances the difficult characteristics of both the *modality* of the Rastrigin function and the *structure* of the double-sphere. Unfortunately, this balance is disappointing. When we run both algorithms on the double-Rastrigin function, we find that the success rates are lower than 3%, regardless of population size. This is because the success rates for the double-Rastrigin function can be decomposed into the success rates of its components. That is, the probability that an algorithm will be successful on the double-Rastrigin is approximately the joint probability that it is successful on the Rastrigin function *and* the probability that it will succeed on the double-sphere.

This is also an incomplete picture because the results presented so far have not used *random restarts*. From a practical point of view, restarts can increase performance because the success probabilities will add. That is, each restart represents an independent event. So, we are not just forced to find a population that balances the characteristics of both the Rastrigin and double-sphere function, we also need to account for the general observation that smaller populations will use fewer evaluations and restart more often.

When we include restarts and allow each algorithm to use $1e7$ evaluations, we still observe low success rates. For example, CSA-ES peaks at $\hat{\omega} \approx 11\%$ with a population of $\lambda = 400$. CMA-ES operating with $\lambda = 300$ yields an expected best of $\hat{\omega} \approx 5\%$.

The results of this section reinforce the notion that an algorithm's success or failure largely depends on its ability to cope with the features of a function. A population size suitable for Rastrigin's function is a poor choice for the double-sphere and vise-versa.

4 Limiting Exploration with Dynamic Populations in CSA-ES

On the double-sphere function, larger populations in CSA-ES (and CMA-ES) tend to pull the mean towards the funnel with the most samples. When the funnels are close in depth, a larger sub-optimal funnel is more likely to have more samples. Smaller populations are less vulnerable to this because less information being sampled. On the double-Rastrigin, we need the best of both worlds: a small population size to drop into a funnel without being pulled towards a larger basin of attraction, and then a large population size to exploit the underlying structure of that particular funnel.

As a proof of concept, we implemented CSA-ES with a dynamic population size that increased as the global step-size decreased. A decrease in step-size indicates a higher level of exploitation. When search is first exploring, it is utilizing a small population size. As it begins to exploit a promising region, increasing the population size will help exploit the underlying funnel structure. The algorithm is identical to CSA-ES in every way except at the end of each generation, we compute a new population size based on a function of the global step-size σ, the initial step-size σ_0, and an upper bound of the population size, λ_{MAX}.

$$\lambda = \lambda_{\text{MAX}} \left(\frac{\sigma}{\sigma_0} - 1 \right)^2$$

We ensure that λ never falls below the default population size, $\lambda_d = 14$, or exceeds the maximum λ_{MAX}, which is an input parameter.

We ran this strategy, which we denote D-CSA-ES, on the 30-dimensional Rastrigin, double-sphere, and double-Rastrigin functions for the same values of λ used in the previous section, except that D-CSA-ES interprets this value as λ_{MAX}. The resulting search strategy is less effective on the Rastrigin function, but operates at a consistent

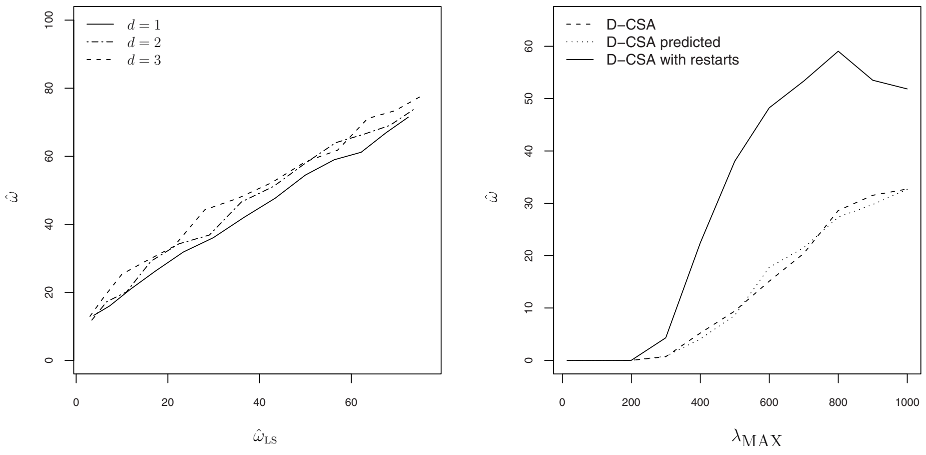


Fig. 5. D-CSA-ES on the double-sphere (left) and on the double-Rastrigin (right). The relationship between success rate and the size of the optimal funnel remains linear. This results in a much higher success rate on the double-Rastrigin function.

level on the double-sphere function that is proportional to the size of the optimal funnel, regardless of the population size. The left graph in Figure 5 shows D-CSA-ES on the double-sphere as a function of optimal funnel size for $d = 1, 2,$ and 3 using $\lambda_{\text{MAX}} = 500$. The most striking feature is the approximately linear relationship between the size of the optimal funnel and the success rate of D-CSA-ES. This resembles the relationship of CMA-ES using a default population size on the double-sphere, but with a setting for λ that is more appropriate for global optimization.

What does this mean for the double-Rastrigin function? The right graph in Figure 5 show D-CSA-ES on the double-Rastrigin function for $s = 0.7$ and $d = 1$ as a function of population size. Without restarts (dash), D-CSA-ES has a success rate the is about 10 times higher than either CMA-ES or CSA-ES. When D-CSA-ES runs with restarts (solid line) until $1e7$ evaluations, it success rates are as high as $\approx 60\%$.

The dotted line in this graph represents the predicted performance obtained by multiplying the $\hat{\omega}$ from Rastrigin with $\hat{\omega}$ from the double-sphere. The prediction is very close to the empirical results and reinforces the notion that successful search must cope with both modality and global structure.

5 Summary

Global structure can clearly impact the performance of evolutionary optimization. When the optimal funnel is proportionally smaller, the success rates for CHC and CMA-ES decrease dramatically on the double-sphere, especially when the depths of the two funnels are close. Exploration is not able to distinguish between funnel quality, and is pulled into the larger funnel. We believe these results generalize to other algorithms.

This presents a problem for CMA-ES and CSA-ES on the double-Rastrigin function because, although larger population sizes are necessary to exploit the underlying structure of the Rastrigin, they are also more bias towards funnel size. The population size that is best for Rastrigin is the least effective on the double-sphere. A compromise that works on both is disappointing.

By dynamically adapting the population size, D-CSA-ES is less biased toward funnel size while exploring the search space. However, as it descends into a particular funnel, and it begins to exploit the search space, increasing the population size aids D-CSA-ES in detecting the underlying structure of the funnel and avoiding local optima. This results in a strategy whose success rate is dependent on funnel size; when the optimal funnel is large, the success rates for D-CSA-ES are not a good as CHC or CMA-ES. But when the optimal funnel is small, D-CSA-ES will still find the global solution with a probability proportional to relative funnel size. The highs are not as high, but the lows are still acceptable.

Exploring the search space to gain a global perspective before exploiting a particular region may be an effective strategy for “big valley”, single-funnel problems. But on multi-funnel landscapes, the effectiveness of exploration comes into question as a global search strategy. This work supports an ongoing awareness that, if an algorithm is going to be successful, then it must be able to deal with the features in the landscape.

Acknowledgments. This research was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number FA9550-07-1-0403. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

1. Eshelman, L.J.: The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In: FOGA (1991)
2. Hansen, N., Kern, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In: PPSN 2004. Springer, Heidelberg (2004)
3. Kern, S., Muller, S., Hansen, N., Buche, D., Ocenasek, J., Koumoustakos, P.: Learning Probability Distributions in Continous Evolutinary Algorithms—a Comparative Review. Natural Computing 3, 77–112 (2004)
4. Leary, R.H.: Global Optimization on Funneling Landscapes. Journal of Global Optimization 18 (2000)
5. Ostermeier, A., Gawelczyk, A., Hansen, N.: Step-Size Adaptation Based on Non-local use of Selection Information. In: PPSN 1994, pp. 189–198. Springer, Heidelberg (1994)
6. Pardalos, P.M., Schoen, F.: Recent Advances and Trends in Global Optimization: Deterministic and Stochastic Methods. In: CAPD (2004)
7. Wales, D.J.: Energy Landscapes and Properties of Biomolecules. Physical Biology (2005)
8. Wales, D.J., Doye, J.P.: Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. Journal of Chemical Physics 101(28) (April 1997)
9. Whitley, D., Beveridge, R., Graves, C., Mathias, K.: Test Driving Three 1995 Genetic Algorithms: New Test Functions and Geometric Matching. Journal of Heuristics (1995)

Improved Lower Limits for Pheromone Trails in Ant Colony Optimization

David C. Matthews

Colorado State University, Computer Science Department
Fort Collins, Colorado, USA 80523
dvmtthws@cs.colostate.edu

Abstract. Ant Colony Optimization algorithms were inspired by the foraging behavior of ants that accumulate pheromone trails on the shortest paths to food. Some ACO algorithms employ pheromone trail limits to improve exploration and avoid stagnation by ensuring a non-zero probability of selection for all trails. The MAX-MIN Ant System (MMAS) sets explicit pheromone trail limits while the Ant Colony System (ACS) has implicit pheromone trail limits. Stagnation still occurs in both algorithms with the recommended pheromone trail limits as the relative importance of the pheromone trails increases ($\alpha > 1$). Improved estimates of the lower pheromone trail limit (τ_{min}) for both algorithms help avoid stagnation and improve performance for $\alpha > 1$. The improved estimates suggest a general rule to avoid stagnation for stochastic algorithms with explicit or implicit limits on exponential values used in proportional selection.

1 Introduction

Metaheuristic search algorithms are an effective means to solve a variety of combinatorial optimization problems [1]. Ant Colony Optimization (ACO) [2] is a group of metaheuristic algorithms inspired by nature. These algorithms model the foraging behavior of ants that accumulate pheromone trails on the best paths leading to food. ACO algorithms similarly build pheromone trails to learn the best solution to a problem.

ACO algorithms perform a series of iterations to explore the problem space while exploiting the pheromone trail information learned from previous iterations. Each iteration involves solution construction using random proportional selection based on the pheromone trails and heuristic information from the problem, local search to improve these solutions, and pheromone trail updates based on one or more constructed solutions. The algorithm ceases at some predetermined point, such as a maximum number of iterations or a time limit.

A common problem with metaheuristic search algorithms such as ACO is stagnation or premature convergence to a local optimum. In ACO, stagnation occurs when a single pheromone trail dominates the random proportional selection during tour construction. ACO algorithms use different methods to address stagnation.

Stützle and Hoos introduced the MAX-MIN Ant System (MMAS) [3] which includes explicit upper and lower pheromone trail limits to avoid stagnation. Dorigo and Gambardella introduced the Ant Colony System (ACS) [4] which includes an exponential moving average pheromone update rule that implicitly limits the pheromone trails.

The recommended pheromone trail limits for both algorithms exhibit stagnation as the relative importance of the pheromone trails increases ($\alpha > 1$). This behavior is consistent with other ACO algorithms and prior literature which resulted in a parameter recommendation of $\alpha = 1$ for MMAS and ACS.

An analysis of the proportional probabilities shows the probability of selection for trails with the minimum pheromone level τ_{min} approaches zero for $\alpha > 1$. The improved estimates of τ_{min} for MMAS and ACS introduced in this article maintain the desired non-zero proportional probabilities to help avoid stagnation.

Benchmark problems from TSPLIB[5] demonstrate the expected stagnation, both with and without local search. The improved estimates of the lower pheromone trail limit τ_{min} reduce the effect of stagnation and significantly improve performance for $\alpha > 1$ on the benchmarks. The improved estimates suggest a general rule for stochastic algorithms with explicit or implicit limits on exponential values used in proportional selection.

In the remainder of this paper we review the estimation of the pheromone trail limits in Section 2, propose improved estimates for lower pheromone trail limits in Section 3, compare empirical results for the existing and improved lower pheromone trail limits in Section 4, and summarize our findings with recommendations for additional additional research in Section 5.

2 Pheromone Trail Limits

ACO algorithms share the first two equations in Table 1 – the probability distribution rule p_{ij}^k used by ants for random proportional selection during tour construction and the pheromone trail update rule τ_{ij} applied at the end of each iteration. Pheromone trail limits τ_{min} and τ_{max} defined in the remaining equations are explicit in MMAS and implicit in ACS. The performance of these algorithms depend on the proper selection of these pheromone trail limits.

During tour construction, random proportional selection based on the probability distribution p_{ij}^k determines the next city j visited by ant k currently located at city i using equation (1). Only cities not yet visited on the tour constructed by ant k are eligible for selection – this is the neighborhood N_i^k. The current pheromone trail values τ_{ij} and the problem heuristics η_{ij} determine the probability distribution, with exponents α and β to weight their relative contributions. ACS omits the α exponent on pheromone trails which is equivalent to setting $\alpha = 1$. Both algorithms may employ pseudo-random proportional selection where the value q_0 biases the amount of exploration versus exploitation of the existing solution.

In MMAS, the pheromone trail update rule for τ_{ij} in equation (2) evaporates a portion of all pheromone trail values based on the evaporation rate ρ and

Table 1. Equations governing the behavior of the MMAS and ACS algorithms

MMAS	**ACS**	
$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta}$	$p_{ij}^k(t) = \frac{[\tau_{ij}(t)] \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)] \cdot [\eta_{il}]^\beta}$	(1)
$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}$	$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}^{bs}$	(2)
$\tau_{min} \leq \tau_{ij} \leq \tau_{max}$		(3)
$\tau_{max} = \frac{1}{\rho} \cdot \frac{1}{L^{opt}}$	$\tau_{max} = \frac{1}{L^{bs}}$	(4)
$\sqrt[n]{p_{best}} = \frac{\tau_{max}}{\tau_{max} + avg \cdot \tau_{min}}$		(5)
$\tau_{min} = \tau_{max} \cdot \left[\frac{(1-\sqrt[n]{p_{best}})}{avg \cdot \sqrt[n]{p_{best}}}\right]$	$\tau_{min} = \frac{1}{L^{nn}} \cdot \frac{1}{n}$	(6)
$\tau_{minlocalsearch} = \tau_{max} \cdot \left[\frac{1}{2 \cdot n}\right]$		(7)
$\tau_0 = \tau_{max}$	$\tau_0 = \tau_{min}$	(8)

deposits additional pheromones on the trail of either the iteration best or global best solution. MMAS also adjusts any pheromone trail values not within the limits τ_{min} and τ_{max} to the appropriate limit to maintain the relationship in equation (3). The result is a relative increase in the probability of selection for some trails during the next iteration and a decrease for others.

In ACS, the pheromone trail update rule in equation (2) applies only to the pheromone trails of the global best solution using an exponential moving average with ρ as the smoothing factor to maintain the pheromone values between the limits. Since the initial pheromone level τ_0 of ACS is the minimum trail level τ_{min} in equation (8), the result is an increase only to trails associated with the best solution. There is no evaporation in ACS.

Both algorithms derive their upper limit τ_{max} in equation (4) from the limit of the series expansion for the pheromone trail update rule $\tau_{ij}(i+1)$, where L^{opt} is the length of the optimal tour and L^{bs} is the length of the global best solution. Since the optimal tour L^{opt} is not known in advance, MMAS substitutes the length of the global best solution during the run L^{bs} as an estimation.

MMAS estimates the lower limit τ_{min} in equation (5) using the probability distribution p_{ij}^k and the probability p_{best} of constructing a greedy solution with the maximum pheromone values[3]. Since the remaining number of selections varies during tour construction, the proportional probability estimation uses the average number of selections remaining. This average is $n/2$ if all n possible selections are considered or $cand/2$ when using a candidate list of size $cand$. Solving the equation for τ_{min} yields equation (6). The initial pheromone level τ_0 for MMAS is set to the maximum pheromone trail level τ_{max} in equation (8) to encourage early exploration. MMAS recomputes the explicit pheromone trail limits during execution as new best solutions are found. Based on experimental results, the recommended value for p_{best} is 0.05. When run with local search, MMAS sets the lower pheromone trail limit τ_{min} using equation (7) to increase exploration[6].

ACS sets an experimentally determined lower limit τ_{min} in equation (6) where L^{nn} is the length of a nearest neighbor tour[4]. In ACS the lower limit does not change from the initial value, while the upper limit changes implicitly as new best solutions are found.

Both lower pheromone trail limits have the form $\tau_{min} = \tau_{max} \cdot a$ where a is a scaling factor between 0 and 1, assuming an equivalence of L^{nn} and L^{bs} in the estimations for ACS. In the next section we will see that the value of the lower pheromone trail limit τ_{min} plays a critical role in the avoidance of stagnation.

3 Improved Lower Limits

An analysis of the proportional probabilities p_{ij}^k using problem kroA100 from TSPLIB[5] as an example suggests stagnation for $\alpha > 1$. The selection probabilities p_{max} and p_{min} for trails with the maximum and minimum pheromone trail values are shown in Table 2. As α increases, the range between τ_{max} and τ_{min} increases and the proportional probabilities for p_{max} approach 1.0 and p_{min} approach 0.0 which results in stagnation.

Table 2. MMAS and ACS selection probability for trails corresponding to maximum p_{max} and minimum p_{min} pheromone trail values using TSP problem kroA100 with $n = 100$. Parameters used in this estimation include $L^{opt} = 21282$, $cand = 20$, $p_{best} = 0.05$, and $\rho = 0.02$ for MMAS, $L^{bs} = L^{nn} = 25848$ for ACS.

α	τ_{max}	τ_{min}	τ_{max}^{α}	τ_{min}^{α}	p_{max}	p_{min}
			MMAS			
1	2.3e-03	7.1e-06	2.3e-03	7.1e-06	0.97	3.0e-03
2	2.3e-03	7.1e-06	5.5e-06	5.1e-11	0.99	9.2e-06
3	2.3e-03	7.1e-06	1.3e-08	3.6e-16	1.00	2.8e-08
4	2.3e-03	7.1e-06	3.0e-11	2.6e-21	1.00	8.6e-11
			ACS			
1	3.9e-05	3.9e-07	3.9e-05	3.9e-07	0.92	9.2e-03
2	3.9e-05	3.9e-07	1.5e-09	1.5e-13	0.99	1.0e-04
3	3.9e-05	3.9e-07	5.8e-14	5.8e-20	1.00	1.0e-06
4	3.9e-05	3.9e-07	2.2e-18	2.2e-26	1.00	1.0e-08

Stagnation occurs for $\alpha > 1$ in MMAS due to the simplified estimate of proportional probabilities used in equation (5) that omitted α, resulting in a derivation for τ_{min} that assumed $\alpha = 1$. Note that the probability of selection $p_{max} = \tau_{max}^{\alpha}/(\tau_{max}^{\alpha} + avg \cdot \tau_{min}^{\alpha})$ and $p_{min} = \tau_{min}^{\alpha}/(\tau_{max}^{\alpha} + avg \cdot \tau_{min}^{\alpha})$ are proportional to τ_{max}^{α} and τ_{min}^{α}, not τ_{max} and τ_{min}. The proposed improvements in Table 3 incorporate α in the proportional probability estimation for p_{ij}^k shown in equation (9). Solving this equation for τ_{min} in equation (10) yields a similar estimation in terms of τ_{max} where the scaling factor is now adjusted by the exponent $\frac{1}{\alpha}$. This narrows the range between τ_{max} and τ_{min} to compensate for the exponentiation in the proportional probabilities. Assuming a general form $\tau_{min} = \tau_{max} \cdot a^{\frac{1}{\alpha}}$ for the lower pheromone trail limit suggests an analogous improvement for MMAS with local search in equation (11) and ACS in equation(10) as shown in the table.

Table 3. Improved lower limits for pheromone trails in MMAS and ACS

MMAS	ACS	
$\sqrt[n]{p_{best}} = \frac{\tau_{max}^{\alpha}}{\tau_{max}^{\alpha} + avg \cdot \tau_{min}^{\alpha}}$		(9)
$\tau_{min} = \tau_{max} \cdot \left[\frac{(1 - \sqrt[n]{p_{best}})}{avg \cdot \sqrt[n]{p_{best}}} \right]^{\frac{1}{\alpha}}$	$\tau_{min} = \frac{1}{L^{nn}} \cdot \left[\frac{1}{n} \right]^{\frac{1}{\alpha}}$	(10)
$\tau_{minlocalsearch} = \tau_{max} \cdot \left[\frac{1}{2 \cdot n} \right]^{\frac{1}{\alpha}}$		(11)

The selection probabilities p_{max} and p_{min} in Table 4 reflect the improved estimates of τ_{min} for both MMAS and ACS. The improved estimates result in constant selection probabilities as α increases for both algorithms. The ACS probabilities will vary somewhat during execution since the value of τ_{max} increases as the value of L^{bs} decreases while the value of τ_{min} remains constant. The MMAS probabilities will maintain their ratio since the algorithm recomputes both when L^{bs} changes.

Table 4. Improved MMAS and ACS selection probability for trails corresponding to maximum p_{max} and minimum p_{min} pheromone trail values using TSP problem kroA100 with $n = 100$. Parameters used in this estimation include $L^{opt} = 21282$, $cand = 20$, $p_{best} = 0.05$, and $\rho = 0.02$ for MMAS, $L^{bs} = L^{nn} = 25848$ for ACS.

α	τ_{max}	τ_{min}	τ_{max}^{α}	τ_{min}^{α}	p_{max}	p_{min}
			MMAS			
1	2.3e-03	7.1e-06	2.3e-03	7.1e-06	0.97	3.0e-03
2	2.3e-03	1.3e-04	5.5e-06	1.7e-08	0.97	3.0e-03
3	2.3e-03	3.4e-04	1.3e-08	3.9e-11	0.97	3.0e-03
4	2.3e-03	5.5e-04	3.0e-11	9.3e-14	0.97	3.0e-03
			ACS			
1	3.9e-05	3.9e-07	3.9e-05	3.9e-07	0.92	9.2e-03
2	3.9e-05	3.9e-06	1.5e-09	1.5e-11	0.92	9.2e-03
3	3.9e-05	8.3e-06	5.8e-14	5.8e-16	0.92	9.2e-03
4	3.9e-05	1.2e-05	2.2e-18	2.2e-20	0.92	9.2e-03

These new estimates for τ_{min} ensure non-zero probability of selection for all trails with $\alpha > 1$ to help avoid stagnation. In the next section, we study the improved stagnation avoidance in MMAS and ACS, both with and without local search.

4 Results

We conducted a series of tests using common TSPLIB[5] benchmark problems to illustrate and verify the expected improvement for the MMAS and ACS algorithms. Optimal solutions known for the TSPLIB problems simplify the analysis of the results, allowing us to illustrate the improvement using the percent over optimal as a measure. We modified the ACO-TSP [7] source code to incorporate the improved lower pheromone trail limit estimates in both algorithms.

Statistical analysis using the Mann-Whitney U test[8] and the Kruskal-Wallis One-Way Analysis of Variance test[8] allow us to characterize the significance of these results. Both tests are non-parametric methods used to test a difference in the location of probability distributions for populations based on the rank of the values. Mann-Whitney tests two populations to determine if the ranks of the combined populations are equivalent in location. Kruskal-Wallis is an extension of Mann-Whitney to test three or more populations to determine if the ranks of the combined populations are equivalent in location. All statistical analysis is based on the average of 10 runs with a statistical significance of 0.05.

The performance results in Table 5 and Table 6 show the percent over the optimal solution for the average of ten runs of 2500 iterations using default parameters and varying both α and β from 1 to 7. Each 7 by 7 grid shows $\alpha = 1$ in the bottom row and $\beta = 1$ in the left column, with white representing optimal and black representing 5.0% over optimal for tests run without local search and 0.2% over optimal for tests run with local search. The MMAS and ACS rows of these tables show results for the original values of τ_{min} while the MMAS+ and ACS+ rows show results for the improved values of τ_{min}.

The pseudorandom proportional rule constant was the recommended $q_0 = 0.9$ for ACS without local search and $q_0 = 0.98$ for ACS with local search. MMAS did not employ this rule. To help avoid stagnation MMAS used pheromone trail reinitialization while ACS used local pheromone trail updates.

For tests run without local search, we characterized the performance for a range of greedy pheromone, heuristic, and evaporation parameter values (α, β, and ρ) on benchmark problems eil51 and kroA100 as shown in Table 5. We selected a range of evaporation rates ρ based on ACO recommendations[2] to study the impact of the new pheromone trail limits on other algorithm parameters.

For tests run with local search, we characterized the performance for a range of greedy pheromone and heuristic parameter values (α and β) on benchmark problems d198, lin318, pcb442, att532, rat783, and pcb1173 as shown in Table 6. The tests employed the 3-opt local search algorithm included in ACO-TSP and the default evaporation rates ρ based on the ACO recommendations[2].

The location of the lowest average score in each grid exhibits different distributions in these tables. For MMAS and ACS the lowest average score occurs in

Table 5. Average performance without local search for the original (MMAS, ACS) and improved (MMAS+, ACS+) lower pheromone trail limits. Each grid shows % over optimal from 0% (white) to 5+% (black) with $\alpha = 1..7$ on the vertical axis, $\beta = 1..7$ on the horizontal axis and $\alpha = \beta = 1$ in the lower left corner.

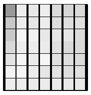
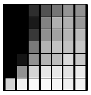
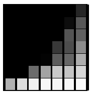
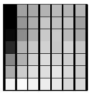
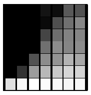
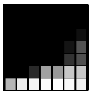
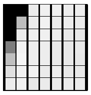
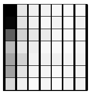
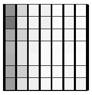
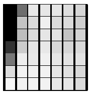
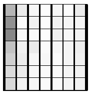
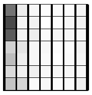
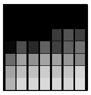
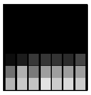
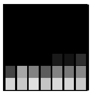
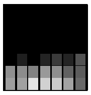
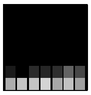
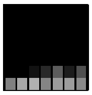
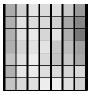
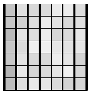
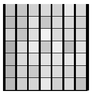
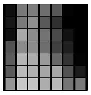
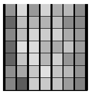
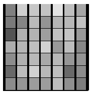
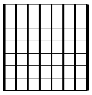
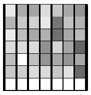
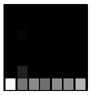
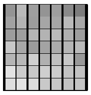
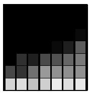
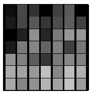
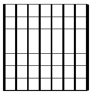
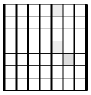
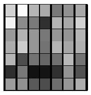
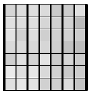
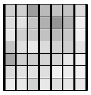
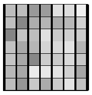
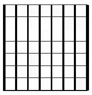
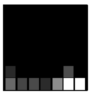
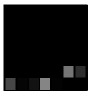
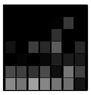
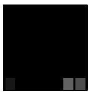
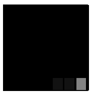
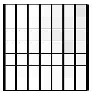
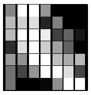
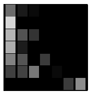
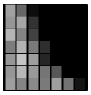
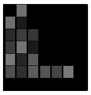
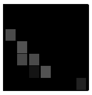
	eil51			**kroA100**		
ρ	0.02	0.2	0.5	0.02	0.2	0.5
MMAS						
MMAS+						
ρ	0.1	0.3	0.5	0.1	0.3	0.5
ACS						
ACS+						

Table 6. Average performance with local search for the original (MMAS, ACS) and improved (MMAS+, ACS+) lower pheromone trail limits. Each grid shows % over optimal from 0% (white) to 0.2+% (black) with $\alpha = 1..7$ on the vertical axis, $\beta = 1..7$ on the horizontal axis and $\alpha = \beta = 1$ in the lower left corner.

	d198	lin318	pcb442	att532	rat783	pcb1173
MMAS						
MMAS+						
ACS						
ACS+						

the $\alpha = 1$ row in 23 of 24 test cases, but in only 2 of 24 test cases for the $\alpha > 1$ rows. For MMAS+ and ACS+ the lowest average score occurs in the $\alpha > 1$ rows in 21 of 24 test cases, but in only 7 of 24 test cases for the $\alpha = 1$ row. In some test cases, the lowest average score appeared in both categories.

The original and improved versions of the algorithms produce the same lower pheromone trail limits τ_{min} when $\alpha = 1$ which should lead to similar performance. The difference in performance was not statistically significant in 23 of 24 tests cases using the Mann-Whitney test.

Both tables appear to exhibit the expected stagnation for MMAS and ACS with darker shading representing increased error as alpha increases. The difference in performance for MMAS and ACS between the $\alpha = 1$ and $\alpha > 1$ populations are statistically significant in 21 of 24 test cases using the Mann-Whitney test, confirming stagnation does occur for $\alpha > 1$ with the original lower pheromone trail limits.

MMAS+ and ACS+ do not appear to exhibit the same stagnation behavior. The difference in performance for MMAS+ and ACS+ between the $\alpha = 1$ and $\alpha > 1$ populations are not statistically significant in 21 of the 24 test cases using the Mann-Whitney test, confirming stagnation does not occur for $\alpha > 1$ with the improved lower pheromone trail limits.

The improved lower pheromone trail limits in MMAS+ and ACS+ also appear to deliver more consistent results across the range of α values tested. The difference in performance for individual values of α is not significant in 19 of 24 tests cases with MMAS+ and ACS+ using the Kruskal-Wallis test. This condition holds for only 4 of 24 test cases with MMAS and ACS.

The test cases run without local search in Table 5 also studied the impact of changes to the evaporation rate ρ to identify possible effects on other algorithm parameters. We compared the performance of the recommended ρ value with the larger values of ρ tested. The difference in performance is statistically significant in 14 of 16 test cases, with 7 of 8 test cases for MMAS and ACS showing decreased performance as ρ increased and 7 of 8 test cases for MMAS+ and ACS+ showing increased performance as ρ increased.

We also compared the performance of MMAS+ and ACS+ for all three tested values of ρ with the best value of ρ for MMAS and ACS. The difference in performance is statistically significant in 11 of 12 test case comparisons using the Mann-Whitney test, with MMAS+ and ACS+ providing better performance.

These results confirm a significant reduction in stagnation and improvement in the performance for $\alpha > 1$ on the problems tested when using the improved lower limit for pheromone trails τ_{min} for both ACO algorithms. These results also demonstrate significant differences in algorithm behavior that result from changes in other parameters with the improved lower limits for pheromone trails.

5 Conclusions

Pheromone trail limits in ACO algorithms are a more effective means of avoiding stagnation than previously understood. Our tests show that performance of these algorithms is sensitive to proper selection of the lower pheromone trail limit. The omission of α in the previous estimates for the lower pheromone trail limit produced values of τ_{min} that assumed $\alpha = 1$, leading to stagnation for larger values of α. The new estimates for the lower pheromone trail limits significantly reduce stagnation and improve performance of both MMAS and ACS on the test problems. This improvement occurs both with and without the use of local search.

The set of tests used in this paper to characterize the improvement are meant to be illustrative of the performance improvement in solution quality only. Additional tests are necessary to study the impact on larger Traveling Salesman Problems and other types of problems such as the quadratic assignment and scheduling problems. Of particular interest is whether the use of greedy pheromones, $\alpha > 1$, with the improved lower pheromone trail limits will improve performance on problems that have relatively poor heuristics.

The parameter testing with ρ shows significant improvement with values larger than the recommended defaults for both algorithms. This constitutes a trade-off between early exploration before convergence with smaller evaporation parameters and exploration near the local optimum after convergence with larger evaporation parameters. Performance for problems with strong fitness-distance correlations such as TSP[3] may benefit more from the latter form of exploration with the improved stagnation avoidance offered by the improved lower pheromone trail limits.

The improvement in MMAS performance was consistent for tests with and without local search. The improvement in ACS performance was less apparent in tests with local search. The higher pseudorandom proportional selection value $q_0 = 0.98$ used by ACS with local search favors the best solution found and may limit exploration near the local optimum. The performance of ACS with local search on these problems may benefit from a lower value of q_0 as a result of the improved stagnation avoidance.

The improved stagnation avoidance from the new lower pheromone trail limits may alter the results of previous studies of ACO parameter optimization[9] [10] [11]. These studies found performance improvements with a range of values of α, ρ, and q_0 on selected problems. Further study may provide an expanded range of values for parameter optimization methods or suggest changes to recommended parameter settings.

Our tests focused on stagnation avoidance and solution quality. We did not study the effect of $\alpha > 1$ on computation time, using 2500 iterations as the stopping criteria for all problems. Additional tests are necessary to understand the impact on computation time for $\alpha > 1$ and compromise in solution quality for a fixed time limit.

Finally, the improved estimation suggests a general approach to avoid stagnation for stochastic algorithms with explicit or implicit limits on the exponential values used in proportional selection.

$$p = \frac{v^e \ldots}{\sum v^e \ldots} \tag{12}$$

$$v(i+1) = (1-\lambda) \cdot v(i) + \lambda \cdot \mathit{\Delta} v \tag{13}$$

$$v_{min} \leq v(i) \leq v_{max} \tag{14}$$

$$v_{min} = v_{max} \cdot f^{\frac{1}{e}} \tag{15}$$

Given a proportional probability of selection p in equation (12) where the value v is modified each iteration using an update rule such as an exponential moving average in equation (13) and the value v is bounded as in equation (14), then the limits should satisfy the relationship in equation (15) where f is some scaling factor between 0 and 1.

Acknowledgments

We thank L. Darrell Whitley, Daniel F. Massey, Charles W. Anderson, and the anonymous reviewers for their valuable comments.

References

[1] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
[2] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
[3] Stützle, T., Hoos, H.H.: Max-min ant system. Future Generation Comp. Syst. 16(8), 889–914 (2000)
[4] Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evolutionary Computation 1(1), 53–66 (1997)
[5] Reinelt, G.: TSPLIB, `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html`
[6] Stützle, T.: Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications. PhD thesis, Technische Universität Darmstadt (1998)
[7] Stützle, T.: ACOTSP, `http://iridia.ulb.ac.be/ mdorigo/ACO/aco-code/public-software.html`
[8] SPSS, Inc.: SPSS 16.0 (2007)
[9] Randall, M.: Near parameter free ant colony optimisation. In: ANTS Workshop, pp. 374–381 (2004)
[10] Gaertner, D., Clark, K.L.: On optimal parameters for ant colony optimization algorithms. In: IC-AI, vol. 1, pp. 83–89 (2005)
[11] Pellegrini, P., Favaretto, D., Moretti, E.: On max-min ant system's parameters. In: ANTS Workshop, pp. 203–214 (2006)

Evolving Neural Networks for Online Reinforcement Learning

Jan Hendrik Metzen[1], Mark Edgington[2], Yohannes Kassahun[2], and Frank Kirchner[1,2]

[1] Robotics Lab, German Research Center for Artificial Intelligence (DFKI)
Robert-Hooke-Str. 5, D-28359, Bremen, Germany
[2] Robotics Group, University of Bremen
Robert-Hooke-Str. 5, D-28359, Bremen, Germany

Abstract. For many complex Reinforcement Learning problems with large and continuous state spaces, neuroevolution (the evolution of artificial neural networks) has achieved promising results. This is especially true when there is noise in sensor and/or actuator signals. These results have mainly been obtained in offline learning settings, where the training and evaluation phase of the system are separated. In contrast, in online Reinforcement Learning tasks where the actual performance of the systems during its learning phase matters, the results of neuroevolution are significantly impaired by its purely exploratory nature, meaning that it does not use (i. e. exploit) its knowledge of the performance of single individuals in order to improve its performance during learning. In this paper we describe modifications which significantly improve the online performance of the neuroevolutionary method Evolutionary Acquisition of Neural Topologies (EANT) and discuss the results obtained on two benchmark problems.

1 Introduction

Reinforcement Learning (RL) is concerned with deciding which action a (virtual or real) agent should take in a given state of an environment in order to maximize its long-term reward. The strategy an agent follows is called its *policy*. Traditionally, methods from the domain of Temporal Difference (TD) Learning [10] have been most popular for solving RL problems. TD learning is essentially a search in value function space where the policy is *indirectly* optimized by changes in estimated value of a state-action pair. In contrast, methods which search directly in the space of policies have gained more attention recently. Examples of these kind of methods are policy gradient [11] and neuroevolution [15]. In *neuroevolution* (NE), the policy an agent follows is represented as an artificial neural network (ANN), which represents a mapping from state to action. NE optimizes an ANN (and thus the policy) by applying an Evolutionary Algorithm (EA), which modifies either the weights or both the topology and the weights of the ANN. Typically, the policy represented by an ANN is not updated after each step as usually done in TD learning but after each episode or after a fixed number of steps. This is due to the fact that EAs usually assess the performance of an individual (in this case the ANN) as a whole using a fitness function. In episodic, stochastic RL problems, it is most natural to use an approximation of the expected reward per episode as fitness function, e. g. by following the given policy for a certain number of episodes and approximate the expected long-term reward by the average of the actually obtained reward per episode. A critical question is for how many episodes a policy is followed before its expected reward per episode is estimated [6].

It has been shown that NE can outperform TD methods in domains with large and/or continuous state and action spaces especially when sensors and/or actuators are subject to noise [2,12]. However, the comparison between TD learning and NE has usually been done only for offline RL. *Offline RL* means that the agent has a training phase where its actual performance does not matter and only after this phase is it confronted with the real problem where it should act optimally. In contrast, in *online RL* the agent has to act as well as it can from the very beginning, meaning that it has to maximize the obtained reward starting from the first step. Online RL is important since not all tasks can be shaped into an offline RL problem. For instance, the dynamics of real world tasks like robot control are often not completely known or too complex to be accurately simulated, and thus an agent would likely need to learn online in the real world.

Online RL is more challenging for NE mainly due to the following reason: At each moment in time the EA used in NE operates on a whole set of individuals (ANNs), the population. This population normally contains a broad range of ANNs and usually only a few of them represent policies that acquire a large long-term reward. In offline RL, at the end of the training phase only the best individual of the entire training is chosen to perform in the testing phase (the phase in which its performance matters), and because of this a large long-term reward in the testing phase is obtained. However, this is not possible in online RL since it is not known beforehand which ANNs represent a good policy, and it is therefore necessary to test bad individuals frequently in order to assess their fitness. This decreases the overall performance of any NE method in online RL drastically.

In this paper, we will discuss how an NE method can be modified such that the online performance of the method is greatly improved. We will first give a review of works which have applied NE in the context of online RL (Section 2), explain the basic EANT algorithm and its extension to online RL (Section 3), and present the results obtained by offline and online EANT as well as by the TD method Sarsa(λ) on two benchmarks (Section 4). We will conclude with a brief outlook (Section 5).

2 Review of Works

In this section, we discuss works in which neuroevolutionary methods have been applied in the context of online RL and the closely related RL with real-time demands. For a general review of the works in the evolution of neural networks we refer to Yao [15].

Whiteson et al. [14] conducted an empirical study on how to balance the trade-off between exploration and exploitation in the context of neuroevolution. In order to transfer mechanisms for balancing exploitation and exploration from TD learning to neuroevolution, the level at which those mechanisms are applied is modified: instead of applying these mechanisms on the level of individual actions, they were applied on the level of episodes, in which entire policies are assessed holistically. This was necessary because evolutionary methods have no notion of the value of individual actions but only of whole policies (i. e. they perform a search in policy space and not in value function space). Three different mechanisms were compared (ϵ-greedy selection, softmax selection, and interval estimation) on two benchmark problems (mountain car [10] and server job scheduling [13]) using the neuroevolutionary method NEAT [7]. It was found that all three mechanisms clearly outperform the offline version of NEAT (where all policies get the same number of evaluations) in terms of maximizing the overall accumulated reward. Furthermore, softmax selection and interval estimation performed roughly the same, but both performed significantly better than ϵ-greedy selection.

Realtime demands in the context of RL are addressed by Stanley et al. [8]. They describe how NEAT can be modified in order to meet the realtime demands of a multi-agent continuous-state machine learning game called NeuroEvolving Robotic Operatives (NERO). The main modification of the NEAT algorithm is that instead of replacing an entire population at once, only single individuals are replaced. The worst performing agent among those members of the population that have been evaluated sufficiently is replaced by a new agent created by NEAT's standard mechanism for producing offspring from a given population. The idea of replacing only one individual at once originates from the area of evolution strategies [1] and is known as *steady-state* evolution.

The main contribution of this work is to show that balancing exploitation and exploration and steady-state evolution are two orthogonal concepts and can be meaningfully combined. Furthermore, a new mechanism for balancing exploitation and exploration that is especially suited for neuroevolution is proposed.

3 Evolutionary Acquisition of Neural Topologies (EANT)

3.1 Offline-EANT

In this section, we discuss the EANT algorithm[1] [3]. In the context of this work, we refer to the algorithm proposed in [3] as *Offline-EANT.*

The basic procedure of Offline-EANT is outlined in Algorithm 1. Its structure is similar to most generational EAs: first, the initial population of *num_indiv* individuals is created. Then, for each generation, the respective population is evaluated for a given (fixed) number of episodes *num_evals*. Based on these evaluations, the fitness of each individual is estimated. In `Create-Next-Generation`,

[1] The source code for EANT is available under the URL *http://sourceforge.net/projects/mmlf/*

```
Offline-EANT(num_evals, num_indiv):
    $population \leftarrow$ Create-Individuals($num\_indiv$)
    while $True$ do
        for $i \in \{0, \ldots, num\_evals\}$ do
            $individual \leftarrow$ Choose-Individual($population$)
            $fitnessSample \leftarrow$ Evaluate-Individual($individual$)
            Update-Estimated-Fitness($individual$,$fitnessSample$)
        $population \leftarrow$ Create-Next-Generation($population$)
```

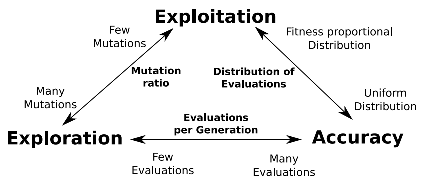
Algorithm 1. The offline (generational) EANT algorithm

some of the members of the population are selected with respect to their fitness and these individuals and their offspring (generated using the genetic operators) form the next population. Creation of the initial population and of the next population based on the current one are described in more detail in Metzen et al. [5]. The individuals EANT acts on are usually encoded in a genome and evaluating them involves developing these genotypes into the corresponding phenotypes. In principle, EANT can be combined with a magnitude of genetic encodings of neural networks. However, in practice EANT has typically been combined with the Common Genetic Encoding (CGE) [4].

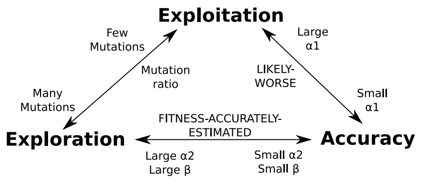
The most important properties of Offline-EANT are (1) that the evolution is divided into two time scales, a smaller one on which the ANN's weights are optimized and a larger one on which the topology of the ANN is optimized [3], and (2) that (optionally) the population is divided into species and the concept of fitness sharing is applied to these species as proposed by Stanley [7].

3.2 Tradeoffs in Offline-EANT

Each evaluation of an individual (policy) in Offline-EANT consists of playing a full episode with this policy, and results in one sample of the fitness function. In deterministic environments with fixed start state, the same individual will always get the same fitness, and thus there is no need to evaluate an individual more than once. In stochastic environments, however, the fitness samples are "noisy" and one evaluation per individual is not sufficient for estimating an individual's fitness accurately. Thus, two questions arise: What is an optimal value for *num_evals* (the evaluations per population) and how should the evaluations be distributed among the individuals? In neuroevolution, it turns out that there is more than just a *exploration-exploitation* trade-off. There is also the issue of estimating an individuals fitness accurately (see Figure 1a). Choosing the value of *num_evals* influences the *exploration-accuracy* trade-off since setting *num_evals* to large values will increase the accuracy of fitness estimates while decreasing the extent to which new network structures and weights are explored, and vice versa. On the other hand, how evaluations are distributed among individuals influences the *exploitation-accuracy* trade-off, since



(a)



(b)

Fig. 1. The three conflicting objectives of neuroevolution (exploitation, exploration, and accuracy), and how they can be controlled by Offline-EANT (a) and Online-EANT (b)

distributing them equally among the individuals will maximize average fitness estimation accuracy but will not exploit information accrued during evaluation, while other distributions will necessarily decrease the estimated fitness accuracy for some individuals. The trade-off between exploitation and exploration is influenced both indirectly via the accuracy and directly via the mutation ratio since less mutations will increase the number of evaluations of individuals which have been found before to perform well (i. e. exploitation) but will decrease the occurrence of new individuals (i. e. exploration), and vice versa. Whiteson et al. [14] (see Section 2) have mainly addressed the exploitation-accuracy trade-off by comparing different ways of distributing the evaluations among the individuals (like softmax selection and interval estimation). These approaches can easily be realized by implementing `Choose-Individual` in Algorithm 1 accordingly. The exploration-accuracy trade-off, on the other hand, could be addressed by manually adjusting *num_evals*. However a systematic, automatic approach would be highly desirable.

3.3 Online-EANT

In this section, we discuss how the EANT algorithm can be modified in a way that allows it to perform more efficiently in online RL tasks. The resulting algorithm, called *Online-EANT*, is summarized in Algorithm 2 and has the form of a *steady-state evolution*, i. e. the evolution is no longer divided into generations. Instead, there is a continuous change in the population, some new individuals being "born", some older individuals "dying" (as in nature). The population is divided into two disjoint sets of individuals. The first set is the set of "mature" individuals. The fitness of these individuals has been accurately estimated, and they are able to create offspring. The other set is the set of "adolescent" individuals. These are the individuals which are currently tested by the algorithm in the environment. None of the adolescent individuals is able to create offspring, and none of the mature individuals will ever be evaluated in the environment again[2].

[2] At this point, it is assumed that the fitness function is stationary, i. e. that the fitness distribution of an individual does not change over time.

```
Online-EANT(num_indiv):
    $adolescents \leftarrow$ Create-Individuals($num\_indiv$)
    $matures \leftarrow \emptyset$
    while $True$ do
        $individual \leftarrow$ Choose-Randomly($adolescents$)
        $fitnessSample \leftarrow$ Evaluate-Individual($individual$)
        Update-Estimated-Fitness($individual$,$fitnessSample$)
        if Likely-Worse($individual$,$matures$) then
            $adolescents.replace$($individual$,Create-New-Offspring($matures$))
        else if Fitness-Accurately-Estimated($individual$) then
            if Fit-Enough($individual$,$matures$) then
                $matures.add$($individual$)
                if To-Many-Individuals($matures$) then
                    $matures.remove\text{-}oldest$()
            $adolescents.replace$($individual$,Create-New-Offspring($matures$))
```

Algorithm 2. The online (steady-state) EANT algorithm

Initially, there are only adolescent individuals (the quantity is determined by the parameter *num_indiv*). For each of these adolescent individuals, there are two possibilities: either they become mature after some time (i. e. evaluations) or they "die" prematurely. Which of these options occurs is determined based on the set of fitness samples obtained by the individual in a way similar to Stagge [6]. For this purpose, two criteria are checked:

(1) If the hypothesis h_1 that the individuals true fitness is below the average estimated fitness of the mature individuals is valid with a certain significance level α_1, the individual is considered to have "died" prematurely and is replaced by a new individual which is created based on the set of mature individuals using EANT's standard procedure of selection and applying genetic operators. The hypotheses h_1 is checked by the function `Likely-Worse`.

(2) If hypothesis h_2 that the individuals current fitness estimate differs from its true fitness by less than a factor β is valid with a certain significance level α_2, it is checked whether the individual's estimated mean fitness is larger than the average fitness of the mature individuals. If that is the case, the individual is added to the mature individuals. If afterwards, the number of mature individuals exceeds a certain number, the oldest (*not* the least fit) mature individual is removed from *matures*. If the individual's estimated mean fitness is not larger than the average fitness of the mature individuals, it is not becoming a mature individual but is replaced by a new individual. The hypotheses h_2 is checked by the function `Fitness-Accurately-Estimated`.

3.4 Handling of Tradeoffs in Online-EANT

As in Offline-EANT (see Section 3.2), there are three conflicting objectives in Online-EANT: exploration, exploitation, and accuracy. In Online-EANT, the exploitation-accuracy and the exploration-accuracy trade-offs are explicitly addressed: (1) The method `Likely-Worse` ensures that no evaluations are "wasted" on individuals which have turned out to be not competitive (on the basis of the obtained fitness samples of the individuals). An individual is considered to be not competitive if the hypothesis h_1 is valid with a certain significance level α_1 (the p-value is computed using an one-tailed, one-sample t-test). The smaller α_1, the more cautious the algorithm is in discarding an individual (i. e. the fitness of an individual needs to be estimated more accurately before it is discarded) but at the expense of exploiting the knowledge of good individuals less. Thus, α_1 controls the exploitation-accuracy trade-off. (2) The method `Fitness-Accurately-Estimated` controls how accurately the fitness of a (competitive) individual is estimated. The fitness is considered to be estimated accurately if the hypothesis h_2 is valid with a certain significance level α_2 (the p-value is computed using a two-tailed, one-sample t-test). The smaller the required significance level α_2 and the allowed error rate β, the more accurately the fitness of an individual is estimated, but at the cost of a reduced amount of exploration. Thus, α_2 and β control the exploration-accuracy trade-off.

The exploitation-exploration trade-off is addressed explicitly in Online-EANT via the mutation rate, but it can be influenced also implicitly via the two other trade-offs: decreasing α_1 and increasing α_2 at the same time will result in roughly the same accuracy but will increase exploitation at the cost of exploration. The relationship between the three issues is depicted in Figure 1b. Whether Online-EANT's way of handling the trade-offs yields better online RL performance than formerly studied approaches is discussed in Section 4.

4 Results

Mountain Car. We have tested three different RL methods in the Mountain Car benchmark [10]: The TD method Sarsa(λ), Offline-EANT, and Online-EANT. For Sarsa, we used the Cerebellar Model Articulation Controller (CMAC) [10] function approximator with a superposition of 10 independent tilings, replacing eligibility traces with decay rate $\lambda = 0.95$, a learning rate of $\alpha = 0.5$, and a discount factor of $\gamma = 1.0$. Two different values for ϵ (the ratio of choosing a random, non-greedy action) have been tested, namely $\epsilon = 0.0$ and $\epsilon = 0.01$. The initial Q-values were all set optimistically to 0 to enforce initial exploration. For Offline-EANT, we used a population size of 20, evaluated each individual for 10 episodes, and disabled fitness sharing. For Online-EANT, we created 20 individuals initially, set the required significance levels α_1 and α_2 to 0.05 and the allowed error rate β to 20%. The maximum number of steps an individual was allowed to take to reach the goal was restricted to 500. If the goal was not reached after this number of steps, the next individual took over control of the car at the current position. Thus, no artificial ending of an episode was necessary and neuroevolution could not get stuck in an unfeasible policy which never reached the goal.

Figure 2a shows the reward per episode of the three methods in the benchmark (the plotted values are the averages over 50 independent runs for each method and are smoothed using a moving window average in order to reduce fluctuations induced by the stochastic start state of the benchmark). Online-EANT obtains significantly more reward per episode than Offline-EANT over the whole 25000 episodes of evaluation ($p < 7 * 10^{-5}$). This shows that Online-EANT's choice of evaluating more promising individuals more often does not only improve initial performance but does also not interfere with the long-term progress of the evolution. Compared to Sarsa ($\epsilon = 0.0$), Online-EANT obtains less reward per episode in the early phase of a trial (the difference is significant during the first 4000 episodes ($p < 0.003$)) but achieves better performance in the long run (the difference is significant after 10000 episodes ($p < 0.05$)). In contrast, Offline-EANT's performance remains significantly worse than Sarsa's ($p < 0.05$) over the whole evaluation time. Sarsa's superior performance for in the initial phase of the trials can be explained by the fact that it makes use of the instantaneous reward supplied by the environment (instead of assessing the policies as a whole) while its worse performance in the long run might be explained by the fact that it is more easily trapped in locally optimal policies (in particular if the learning rate ϵ is set to 0). However, setting ϵ to 0.01 did not yield in an improved online performance (see Figure 2a). Compared to the results presented in [14], Online EANT achieves better online performance than the combination of NEAT with any policy selection strategy. However, since Offline-EANT achieves better performance than the performance reported for Offline-NEAT in [14], this gives no indication on whether the proposed online modifications for EANT are better than different policy selection strategies. Because of this, we analyze this issue in the next section on a more selective problem.

RoboCup Keepaway. Keepaway is part of the RoboCup Soccer Simulator and was introduced as a benchmark by Peter Stone et al. [9]. Metzen et al. [5] have shown that the combination of Offline-EANT and CGE can outperform the results which have been published for the temporal difference learning method Sarsa(λ) and for the neuroevolutionary method NEAT [12] in the Keepaway benchmark problem when given enough training and applied in an offline scenario. In contrast to this offline assumption, we compare in this section the *online* performance of Online-EANT with two versions of traditional, generational EANT (Offline-EANT). For Online-EANT, we created 50 individuals initially and set the maximal number of mature individuals to 25. We set α_1 and α_2 to 0.05 and β to 10%. For Offline-EANT, we used a population size of 50. We compared two different strategies for balancing exploitation and estimation accuracy in Offline-EANT, namely giving all individuals the same number of evaluations and applying softmax policy selection like proposed by Whiteson [14]. The number of episodes per generation was set to $num\_evals = 2000$, giving each individual 40 evaluations in the average. For both Online- and Offline-EANT, we enabled fitness sharing.

Figure 2b shows the performance of the three methods (the plotted values are the averages over 6 independent runs for each method and are smoothed using a moving window average in order to reduce fluctuations induced by the stochastic start state of the benchmark). Online-EANT's performance is

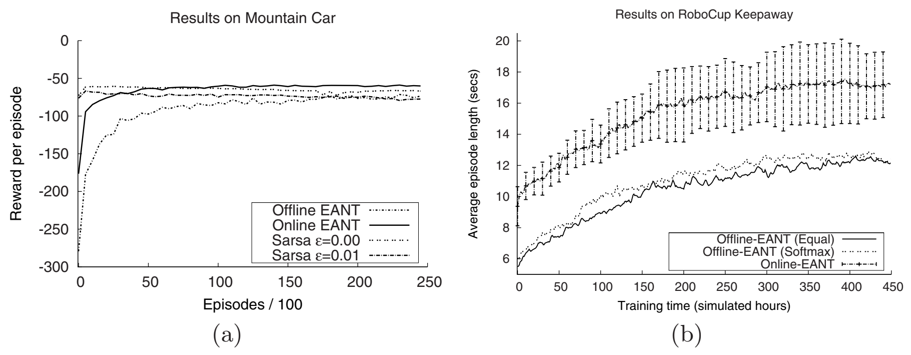


Fig. 2. Performance Comparison on the Mountain Car (left) and RoboCup Keepaway (right) benchmark. Each curve is an average over 50 (Mountain Car) and 6 (Keepaway) independent runs.

significantly better than Offline-EANT's ($p < 0.009$) and Offline-Softmax-EANT's performance ($p < 0.05$) during the whole $450h$ of evaluation. In contrast, Offline-Softmax-EANT's performance is only in the first 6 hour significantly better than Offline-EANT's ($p < 0.05$). Altogether, the results show that proposed method, Online-EANT, improves the online performance of EANT significantly while simply using Offline-EANT with a modified policy selection strategy like Softmax policy selection does not (or only to a very small amount which could not be detected in 6 runs).

5 Conclusions and Outlook

In this paper, a method of extending the neuroevolutionary method EANT has been proposed that permits efficient learning in online RL tasks. In terms of maximizing the accumulated reward, this extension outperforms previous approaches in which softmax policy selection is applied to classical offline neuroevolution. The discussed method mainly addresses the issue of maximizing the accumulated reward during learning. However, for applying RL algorithms in general and neuroevolution in particular to online RL, further issues need to be addressed: when learning in the real world, some actions might be dangerous to the system in certain situations. For instance, when learning to control a helicopter, it must be assured that no policy will explore actions that lead to a crash of the system (which would have virtually infinite costs). In this case, some kind of domain knowledge (an approximate model, for instance) needs to be incorporated into the method. Furthermore, it would be interesting to investigate how the proposed method performs in environments with different characteristics, e. g. environments with a non-stationary fitness function.

References

1. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies - a comprehensive introduction. Natural Computing: an International Journal 1(1), 3–52 (2002)
2. Gomez, F.J., Schmidhuber, J., Miikkulainen, R.: Efficient non-linear control through neuroevolution. In: Proceedings of the 17th European Conference on Machine Learning (ECML), Berlin, Germany, September 2006, pp. 654–662 (2006)
3. Kassahun, Y.: Towards a Unified Approach to Learning and Adaptation. PhD thesis, Institute of Computer Science and Applied Mathematics, Christian-Albrechts University, Kiel, Germany (February 2006)
4. Kassahun, Y., Edgington, M., Metzen, J.H., Sommer, G., Kirchner, F.: A common genetic encoding for both direct and indirect encodings of networks. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1029–1036 (2007)
5. Metzen, J.H., Edgington, M., Kassahun, Y., Kirchner, F.: Analysis of an evolutionary reinforcement learning method in a multiagent domain. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008), Estoril, Portugal, pp. 291–298 (May 2008)
6. Stagge, P.: Averaging efficiently in the presence of noise. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 188–200. Springer, Heidelberg (1998)
7. Stanley, K.O.: Efficient Evolution of Neural Networks through Complexification. PhD thesis, Artificial Intelligence Laboratory. The University of Texas at Austin., Austin, USA (August 2004)
8. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Real-time neuroevolution in the nero video game. IEEE Trans. Evolutionary Computation 9(6), 653–668 (2005)
9. Stone, P., Kuhlmann, G., Taylor, M.E., Liu, Y.: Keepaway soccer: From machine learning testbed to benchmark. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 93–105. Springer, Heidelberg (2006)
10. Sutton, R., Barto, A.: Reinforcement Learning. An Introduction. MIT Press, Massachusetts (1998)
11. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems 12, pp. 1057–1063 (1999)
12. Taylor, M.E., Whiteson, S., Stone, P.: Comparing evolutionary and temporal difference methods in a reinforcement learning domain. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 1321–1328 (2006)
13. Whiteson, S., Stone, P.: Evolutionary function approximation for reinforcement learning. Journal of Machine Learning Research 7, 877–917 (2006)
14. Whiteson, S., Taylor, M.E., Stone, P.: Empirical studies in action selection with reinforcement learning. Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems 15(1), 33–50 (2007)
15. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)

Costs and Benefits of Tuning Parameters of Evolutionary Algorithms

Volker Nannen, S.K. Smit, and A.E. Eiben

Vrije Universiteit Amsterdam
vnannen@gmail.com,{sksmit,gusz}@few.vu.nl

Abstract. We present an empirical study on the impact of different design choices on the performance of an evolutionary algorithm (EA). Four EA components are considered—parent selection, survivor selection, recombination and mutation—and for each component we study the impact of choosing the right operator, and of tuning its free parameter(s). We tune 120 different combinations of EA operators to 4 different classes of fitness landscapes, and measure the cost of tuning. We find that components differ greatly in importance. Typically the choice of operator for parent selection has the greatest impact, and mutation needs the most tuning. Regarding individual EAs however, the impact of design choices for one component depends on the choices for other components, as well as on the available amount of resources for tuning.

1 Introduction

Evolutionary Algorithms (EA) form a class of search methods that work by incrementally improving the quality of a set of candidate solutions by variation and selection [5]. The most important *components* of EAs are thus recombination and mutation (umbrella term: variation), parent selection, and survivor selection. To obtain a working EA, each component needs to be instantiated by a specific *operator*, e.g., the one-point crossover operator for the recombination component. Furthermore, an EA has *parameters* that need to be instantiated by a specific *parameter value*, e.g., 0.5 for the crossover rate. In this paper we maintain the distinction between components and parameters and say that the instantiation of EA components by concrete operators specifies a particular EA, e.g., uniform crossover, bit-flip mutation, random uniform parent selection and k-tournament survivor selection. Further details regarding the parameters do not lead to a different EA, only to variants of the one defined by the operators.[1] A complete EA design includes the definition of an EA (operators for its components) and the specification of a particular variant of it (values for its parameters).

Setting EA parameters is commonly divided into two cases, parameter tuning and parameter control [3,4]. In case of parameter control the parameter values are changing during an EA run. This requires initial parameter values and suitable

[1] Alternatively, components & operators could also be called symbolic parameters & values, and we could say these values only define different EA variants.

control strategies, which in turn can be deterministic, adaptive or self-adaptive. The problem of parameter tuning is hard because for any given application there is a large number of options, but only little knowledge about the effect of EA parameters on EA performance. EA users mostly rely on conventions (mutation rate should be low), ad hoc choices (why not use uniform crossover), and experimental comparisons on a limited scale (testing combinations of three different crossover rates and three different mutation rates). Here we address the problem of parameter tuning. Our main research questions are:

1. How does the choice of operator for each component contribute to EA performance? To this end we compare the absolute performance achieved with different combinations of operators.
2. The parameters of which EA component need the most tuning? For this question we measure the amount of information needed to tune the free parameter(s) of each operator (e.g., crossover rate or tournament size).

For a systematic exploration of the space of EA configurations we use exhaustive search for the combination of operators, and Relevance Estimation and Value Calibration (REVAC) to tune the free (numeric) parameters. REVAC is an Estimation of Distribution Algorithm [14] that tunes an EA by optimizing marginal probability distributions over the free parameters [16,15]. Starting from a set of uniform distributions and an initial drawing of 100 vectors of random parameter values, REVAC iteratively generates new marginal distributions of increasing expected EA performance by *drawing* a new vector of parameter values from the current distributions, *evaluating* the vector by measuring the performance of the EA with these values, *updating* all marginal distributions based on this evaluation, and *smoothing* the updated distributions. Smoothing is a unique feature of REVAC that forces all marginal distributions to approach the maximum Shannon entropy distribution for a given EA performance. This maximized Shannon entropy is independent from the computational cost of any particular tuning method and can be used as a general estimator of the minimum amount of information required to reach a certain level of EA performance. Hence, it can be regarded as a general indicator of how difficult it is to tune a certain EA parameter, and how relevant it is to overall EA performance.

Related work includes the general discussion of EA design [2] and parameter setting [12], in particular within parameter tuning as defined in [3,1,4]. Throughout the relevant literature we find that the costs of tuning parameters is largely ignored. Notable exceptions are the theoretical considerations of [17] and [9], as well as the systematic parameter sweeps of [11,21,20] and the statistical analysis of parameters by [6]. In the general field of experimental design, a paradigm shift that emphasizes a low cost of tuning over the performance of optimal parameter values was due to [22]. In our field, [7] proposes a meta-GA approach in which both EA components and EA parameters are tuned and shows the importance of the right choice for the GA operators. [20] shows how parameter sweeps can be used for robustness and correlation analysis. [18] embed sequential parameter optimization in a wider framework of experimental EA design.

2 Experimental Setup

For a clear discussion we distinguish three different layers in the analysis of an EA: the problem/application (here: fitness landscapes created by a generator), the problem solver (here: an EA), and the method for tuning the problem solver (here: REVAC). For an unbiased study we use independent software implementations for each layer and combine them through simple interfaces. For the problem layer we use a generator of real-valued fitness landscapes that are formed by the max-set of Gaussian curves in high dimensional Cartesian spaces [8]. Where a Gaussian mixture model takes the average of several Gaussians, a max-set takes their enveloping maximum, giving full control over the location and height of all maxima. For the implementation we followed [19] on rotated high dimensional Gaussians, and used 10 dimensions, 100 Gaussians, and the same distributions over height, location, and rotation of these Gaussians as specified in the exemplary problem sets 1–4 of [8]. These sets offer an increasing amount of exploitable structure to the EA. Set 1 has the least structure, with peaks of different height scattered at random, while set 4 is the most structured, with peaks that get higher the closer they get to the origin. For each set, different landscapes are created by passing a different random seed to the generator. Initialization of all EA populations is uniform random in the domain of the fitness landscapes. The optimal fitness value is 1 on each problem instance and the condition for successful termination is defined as "fitness > 0.9999" or 10,000 fitness evaluations".

For the EAs we use the Evolutionary Computation toolkit in Java (ECJ) [13], which allows the specification of a fully implemented EA through a simple parameter file, including the choice of operator for each component and the values for the free parameters. The ECJ offers several operators for each EA component, cf. Table 1. For any given EA, the population size parameter is always present. Most operators have zero or one free parameter. One operator has 2 free parameters—Gaussian(σ, p) with parameters σ for step size and p for mutation probability, which takes the value 1 in case of Gaussian$(\sigma, 1)$. Due to technical details of the ECJ, only 10 different combinations of parent and survivor selection operators are possible.[2] With 4 operators for recombination and 3 operators for mutation, we have 120 combinations of operators, of which 6 with 2, 33 with 3, 53 with 4, 25 with 5, and 3 with 6 free parameters.

The performance of an EA with a given set of parameter values is measured in three different ways: SR (Success Rate, percentage of runs with fitness > 0.9999), MBF (Mean Best Fitness of all runs), and AES (Average number of Evaluations to Solution of successful runs; undefined when SR $= 0$). Each EA is tuned 5 times on each of the 4 problem sets. During each tuning session on a given set REVAC generates 1,000 different vectors of parameter values. Each vector of values is written to the ECJ configuration file, together with the specification of the operators and the problem generator. The resulting EA is evaluated on 10 different instances of the problem set, generated by different random seeds.

[2] Arguably, (μ, λ) and $(\mu + \lambda)$ define both parent *and* survivor selection. Here we classify them under survivor selection because that is what the parameter λ influences.

Table 1. EA components, operators, and parameters used in this study

Component	**Operator**	**Parameter(s)**
		population size μ
parent selection	tournament	parent tournament size
	best selection	number n of best
	random uniform	-
	fitness proportional	-
survivor selection	generational	-
	tournament	survivor tournament size
	random uniform	-
	(μ, λ)	λ
	$(\mu + \lambda)$	λ
recombination	none	-
	one-point	crossover probability
	two-point	crossover probability
	uniform	crossover probability
mutation	reset (random uniform)	mutation probability
	Gaussian$(\sigma, 1)$	step size
	Gaussian(σ, p)	step size, mutation probability

Notes. We follow the naming convention of the ECJ.

For each REVAC tuning session and each EA, the performance after n evaluations is the best performance measured after evaluating n vectors of parameter values. The average performance after n evaluations is averaged over multiple tuning sessions on the same EA. We define *near best performance* as the average performance after 1,000 evaluations minus 5%. If n is the lowest number of vectors for which the average performance exceeds this value, then we say that REVAC needs n evaluations to tune the EA to near best performance. Section 3 uses this to study the impact of choosing an operator for each component.

In Section 4 we analyze the cost and benefits of tuning per EA component. REVAC continuously maximizes the Shannon entropy of the marginal distributions that it optimizes during a tuning session. This maximized Shannon entropy provides a generic information-theoretic measure of the minimum amount of information needed per parameter to reach a given performance level. The differential Shannon entropy H of a probability density function D over the continuous interval $[a, b]$ is commonly defined as

$$H(D_{[a,b]}) = -\int_a^b D(x) \log_2 D(x)\, dx.$$

The sharper the peaks of a probability density function, the lower its Shannon entropy. In order to compare the entropy of distributions that are defined over different parameter intervals in a meaningful way, we normalize all parameter intervals to the interval $[0, 1]$ before calculating the Shannon entropy. In this way the initial uniform distribution has a Shannon entropy of zero, and any other distribution has a negative Shannon entropy $H(D_{[0,1]}) < 0$.

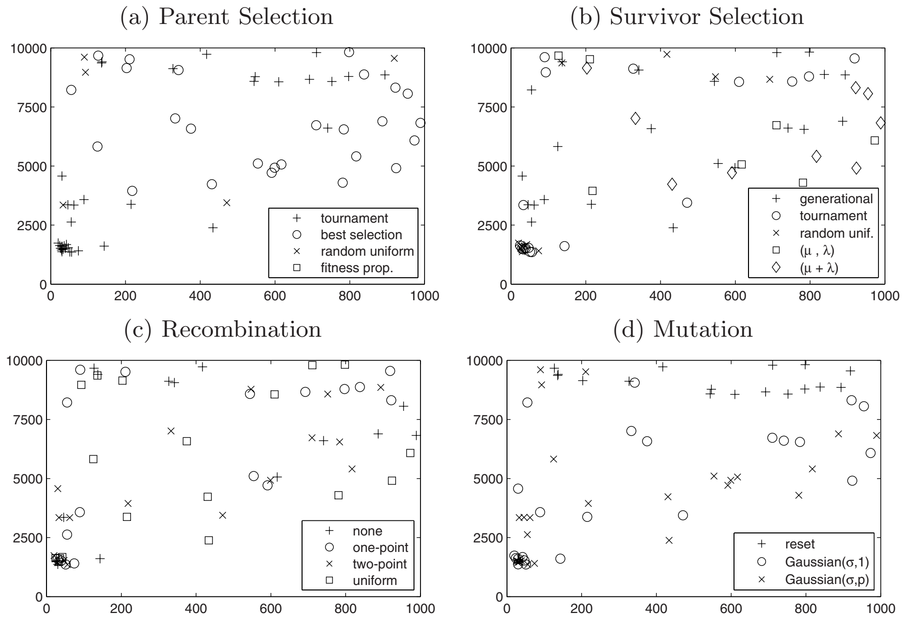


Fig. 1. Near best performance in AES against cost of tuning, by EA component

3 How Does the Choice of Operator Per Component Contribute to Performance?

Due to space limitations we only present data on one problem set (no. 4) and one performance measure. We choose to report on the AES, because it only yields 67 data points (those 67 EAs with SR > 0 for which the AES could be calculated). MBF and SR require 120 data points, making the plots less transparent. The four scatter plots in Figure 1 show the performance of these 67 EAs after tuning, and the cost of tuning, averaged over 5 tuning sessions per EA. The y-axes show the near best performance in AES. The x-axes show the number of REVAC evaluations needed to tune the EA to this performance. Each plot shows the same EAs but labels them according to the operator choice for a different component. To read the full specification of an EA, one needs to look at the same location in all four plots. Table 2 shows the near best performance in AES per operator, averaged over those EAs that have this operator and terminated with success.

The choice of operator for the parent selection component has the strongest effect on EA performance. The 16 EAs that are clustered together in the lower left of each plot of Figure 1 display the best performance and the lowest number of evaluations needed to reach this performance. These EAs all use tournament selection for parent selection, either tournament selection or random uniform selection for survivor selection, any recombination operator, and either Gaussian(σ, p) or Gaussian($\sigma, 1$) for mutation. On the other hand, those 53 EAs that never terminated with success share one common feature, namely a lack

Table 2. Average near best performance in AES per operator

parent selection		survivor selection		recombination		mutation	
random unif.	9581	random unif.	7039	none	7994	Gaussian(σ, p)	6056
tournament	4514	tournament	6332	one-point	7736	Gaussian($\sigma, 1$)	6891
best select.	7661	generational	8299	two-point	7053	reset	9633
fitness prop.	-	(μ, λ)	7943	uniform	7325		
		$(\mu + \lambda)$	7386				

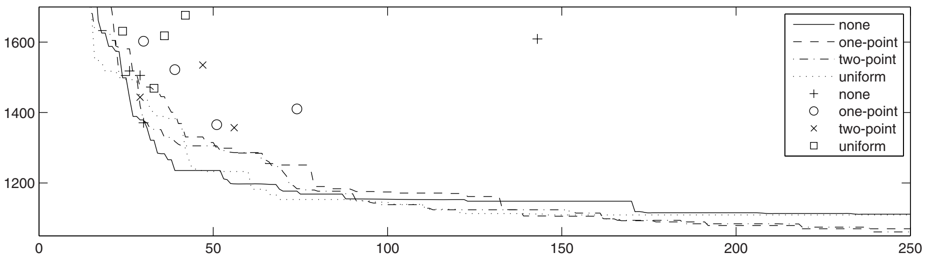


Fig. 2. Impact of recombination operators on AES and cost of tuning

of selection pressure. In particular, EAs with random uniform or fitness proportional selection for parent selection almost never terminate with success unless combined with strong survivor selection pressure.

Of the two variation components, the choice of mutation operator has the stronger effect on EA performance, as can be seen from the differences in Table 2. On this problem set reset mutation is the worst mutation operator, and non-standard Gaussian(σ, p) mutation is superior to Gaussian($\sigma, 1$) both in terms of performance and cost of tuning. The latter may come as a surprise, since the additional free parameter for mutation probability increases the parameter search space. We conclude that the tuning cost of different operators is not additive, and that the tuning cost of an operator can only be evaluated in the context of the overall EA composition.

While choosing the recombination operator has the least effect on EA performance, it demonstrates how the choice of operator can depend on the available resources for tuning. Figure 2 enlarges the lower left corner of Figure 1c, overlaid by four graphs that show the evolution of the average performance of 4 EAs with tournament selection for both parent and survivor selection, Gaussian(σ, p) mutation, and four different recombination operators. 20 tuning sessions were used for each graph. While any recombination operator eventually outperforms no recombination, an EA with no recombination consistently outperforms EAs with recombination after about 30–40 parameter vectors have been evaluated, and it has at least average performance for anything under 100 evaluated parameter vectors. Recombination behaves similar over a wide range of operator choices for the other components and over all 4 problem sets. All in all, for recombination, the choice of operator can clearly depend on the amount of effort that can be invested in tuning.

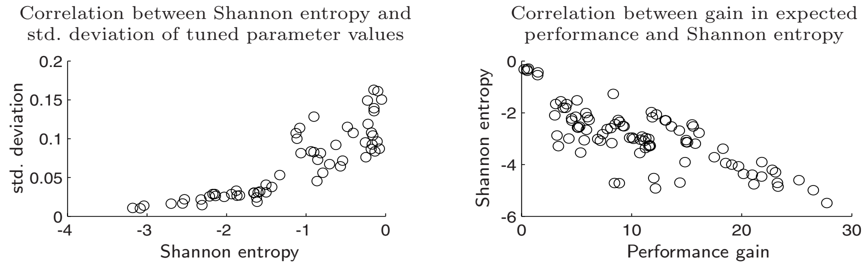


Fig. 3. Correlations with Shannon entropy

4 Which EA Component Needs the Most Tuning?

The previous section related the performance of the near best parameter vector to the number of REVAC evaluations needed to find this vector and to achieve this performance. This section takes a rather unconventional approach based on the average performance when parameter values are drawn from a probability distribution, namely those created by REVAC after 500 evaluations. To calculate the performance gain achieved by tuning, this average performance is compared to the average EA performance when parameter values are drawn from the uniform distribution. All results are averaged over 5 REVAC tuning sessions of an EA on each of the 4 problem sets, 20 tuning sessions per EA. In order to extend our analysis to all 120 EAs, we use the Mean Best Fitness that an EA achieves at termination (successful or not), rather than the AES.

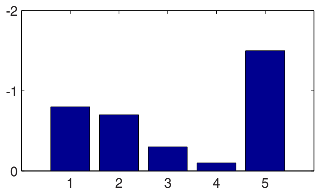
Shannon entropy measures the amount of information that a probability distribution provides on its random values. By definition, the lower the Shannon entropy of the maximum entropy distribution that achieves a given expected EA performance, the finer the parameter value has to be tuned in order to achieve that expected performance. This is demonstrated in Figure 3. The left scatter plot shows the correlation between the Shannon entropy of the marginal distribution over the mutation probability and the standard deviation of the best found parameter values. The x-axis shows the Shannon entropy as estimated by REVAC. The y-axis shows the average of the standard deviation of the 5 best found values for each set. The correlation coefficient is 0.8. The point here is that if the maximum entropy distribution has a higher Shannon entropy, there is less certainty on the precise parameter value, something that can otherwise be expensive to assess.

The right scatter plot of Figure 3 shows a clear correlation between a gain in expected MBF and the Shannon entropy of the maximum entropy distributions that REVAC has estimated after 500 evaluations. The x-axis shows the average performance gain in percent. The y-axis shows the Shannon entropy of the estimated distributions, summed over all tuneable parameters of the EA. Note that no EA lies above the main diagonal, which shows that there is a minimum information cost for every percent point of increase in expected performance,

Table 3. Entropy per EA component & population size aggregated over all EAs

Component & pop. size	**Correl. with MBF gain**		**Shannon Entropy**		
	Correl.	**p-value**	**max**	**median**	**min**
1) pop. size	-0.3	0.002	0	-0.8	-1.5
2) parent sel.	-0.3	0.069	0	-0.7	-3.7
3) surviv. sel.	-0.5	0.002	0	-0.3	-1.2
4) recombin.	-0.3	0.004	0	-0.1	-1.0
5) mutation	-0.6	0	-0.2	-1.5	-4.6
entire EA	-0.8	0	-0.3	-2.9	-5.1

Median Sha. Entropy per Component & pop. size



regardless of the EA specifications. Of those EAs that lie significantly below the diagonal, most use tournament selection for both parent and survivor selection. By 500 REVAC evaluations, their MBF had long been maximized. Further tuning only improved their AES, distorting their performance gain-entropy ratio.

Does the strong correlation between total Shannon entropy and the gain in expected performance carry over to individual EA components? The first two numeric columns of Table 3 show the correlation coefficient for each component and its p-value, i.e., the probability to observe this value if the true coefficient is zero. Only EAs with a tunable operator were considered for the respective component. The correlation is generally weaker, with coefficients up to -.3. In other words, the question which component needs tuning in order to improve the performance of a particular EA depends much on the EA in question.

With respect to the average Shannon entropy per component, we see that not all components require the same amount of tuning. The right numeric columns in Table 3 show the maximum, median, and minimum Shannon entropy that we observed for each component (and the population size) when instantiated with an operator that needs tuning. The bar diagram to the right of Table 3 allows a visual comparison of this average median Shannon entropy. Such a skewed distribution of a need for tuning is commonly known as *sparcity of effects*.

Typically, mutation requires the highest amount of tuning, and recombination the least. This rule has many exceptions, as can be concluded from the low correlation coefficients. While the relative order of Shannon entropy per component depends much on the EA in question, consistent patterns can be detected for small groups of EAs. Take for example the two EAs with tournament selection for both parent and survivor selection, Gaussian$(\sigma, 1)$ mutation and either one-point, two-point or uniform crossover. We find that the Shannon entropy for mutation has the unusually high Shannon entropy of around -.2, while the parent selection operator has a low Shannon entropy below -3. When combining the same selection operators with other recombination or mutation operators, we find that the Shannon entropy for parent selection is back to normal levels, while it is still comparatively high for mutation. Another example is recombination, which only exhibits a low Shannon entropy for uniform crossover in combination with either $(\mu + \lambda)$, or (μ, λ). Such irregular patterns are consistent over different problem sets and seem to be inherent to specific combinations of EA components.

5 Conclusions and Further Work

This paper introduces a novel approach to EA design that emphasizes the cost of tuning. To understand how this cost depends on the choice of operator per EA component, we combined an enumerative search over operators with REVAC for tuning their parameters. Our experiments revealed a number of notable insights.

Our tests confirmed the common wisdom that the choice of operator for one EA component depends on the choice of operator for the other components. Of all components, the choice of operator for parent selection has the biggest impact on EA performance. Furthermore, EAs differ greatly in the amount of tuning needed to reach a given performance, and this tuning cost depends on the overall setup of the EA, rather than the number of free parameters. With regard to recombination, we found that the best EA setup depends on the time and effort one can permit to tune the EA.

To measure the need for tuning per component we use the Shannon entropy of maximum entropy distributions as estimated by REVAC, which expresses the minimum amount of information that is needed to achieve a given expected EA performance. It is a generic information-theoretic measure that is independent of any particular tuning algorithm. Inspired by theoretical considerations, it was validated by a strong correlation with the standard deviation of best solutions found during multiple tuning sessions. Based on this measure we observed that the need for tuning follows a skewed distribution, and that while total Shannon entropy is strongly correlated with performance gain, the correlation per component is weak. The question which component needs the most tuning depends on the precise composition of an EA and can not be answered on a general level. It needs to be addressed by the operational analysis of individual EAs. Also, we recommend that a scientific discussion of individual operators addresses its effect on the overall tunability of an EA and on the need for tuning per component.

Regarding the scope of our results, an empirical study can only use a limited set of test problems, and strictly speaking our findings are only proven for our test problems. However, we consider it unlikely that the complex picture that has emerged here is an artefact of the test problems. What remains to be studied is whether the way in which the need for tuning per component depends on the choice of operators is different on other complex fitness functions.

Last but not least, this paper serves as a demonstration of an open source tool kit that can be used to analyze the need for tuning of EA parameters on a given application. Further documentation, Matlab implementations and graphical demonstrations of REVAC are available on the web sites of the authors[3].

[3] http://www.few.vu.nl/∼volker/revac and http://www.few.vu.nl/∼gusz

References

1. Birattari, M.: The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective. PhD thesis, Université Libre de Bruxelles (2004)
2. Czarn, A., MacNish, C., Vijayan, K., Turlach, B.A., Gupta, R.: Statistical Exploratory Analysis of Genetic Algorithms. IEEE Trans. Evol. Comp. 8(4), 405–421 (2004)
3. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. IEEE Trans. Evol. Comput. 3(2), 124–141 (1999)
4. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter Control in Evolutionary Algorithms. In: Lobo, et al. (eds.) [12], pp. 19–46
5. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
6. François, O., Lavergne, C.: Design of Evolutionary Algorithms—A Statistical Perspective. IEEE Trans. Evol. Comput. 5(2), 129–148 (2001)
7. Friesleben, B., Hartfelder, M.: Optimization of Genetic Algorithms by Genetic Algorithms. In: Albrecht, R.F., Reeves, C.R., Steele, N.C. (eds.) Artificial Neural Networks and Genetic Algorithms, pp. 392–399. Springer, Heidelberg (1993)
8. Gallagher, M., Yuan, B.: A General-Purpose Tunable Landscape Editor. IEEE Trans. Evol. Comput. 10(5), 590–603 (2006)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Boston (1989)
10. Grefenstette, J.J.: Optimization of Control Parameters for Genetic Algorithms. IEEE Trans. Syst. Man Cybernet. 16(1), 122–128 (1986)
11. De Jong, K.A.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)
12. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence. Springer, Heidelberg (2007)
13. Luke, S., et al.: A Java-based Evolutionary Computation Research System, `http://www.cs.gmu.edu/~eclab/projects/ecj/`
14. Mühlenbein, H., Höns, R.: The Estimation of Distributions and the Minimum Relative Entropy Principle. Evolutionary Computation 13(1), 1–27 (2005)
15. Nannen, V., Eiben, A.E.: Efficient Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In: IEEE Congress on Evolutionary Computation (CEC), Piscataway, NJ, USA. IEEE Press, Los Alamitos (2007)
16. Nannen, V., Eiben, A.E.: Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In: Veloso, M.M., et al. (eds.) Proc. of the 20th Int. Joint Conf. on Artif. Intell., IJCAI 2007, pp. 1034–1039. AAAI Press, Menlo Park (2007)
17. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In: Grefenstette, J.J. (ed.) Proc. of the 2nd Int. Conf. on Genetic Algorithms on Genetic algorithms and their application, pp. 224–230. L. E. Associates (1987)
18. Preuss, M., Bartz-Beielstein, T.: Sequential Parameter Optimization Applied to Self-adaptation for Binary-coded Evolutionary Algorithms. In: [12], pp. 91–119
19. Rudolph, G.: On Correlated Mutations in Evolution Strategies. In: Männer, R., Manderick, B. (eds.) Proc. of the 2nd Conf. on Parallel Problem Solving from Nature, pp. 107–116. Springer, Heidelberg (1992)

20. Samples, M.E., Byom, M.J., Daida, J.M.: Parameter Sweeps for Exploring Parameter Spaces of Genetic and Evolutionary Algorithms. In: [12], pp. 161–184
21. Schaffer, J.D., Caruana, R.A., Eshelman, L.J., Das, R.: A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In: Schaffer, J.D. (ed.) Proc. of the 3rd Int. Conf. on Genetic Algorithms, pp. 51–60. Morgan Kaufmann, San Francisco (1989)
22. Taguchi, G., Wu, Y.: Introduction to Off-Line Quality Control. Central Japan Quality Control Association, Nagoya, Japan (1980)

Cooperation in Co-evolving Networks: The Prisoner's Dilemma and Stag-Hunt Games

Enea Pestelacci and Marco Tomassini

Information Systems Department, HEC, University of Lausanne, Switzerland
{enea.pestelacci, marco.tomassini}@unil.ch

Abstract. Interactions giving rise to dilemmas are widespread in society. Starting from the observation that individuals interact through networks of acquaintances, we study the co-evolution of the agents' strategies and of the social network itself using two prototypical games: the Prisoner's Dilemma and the Stag Hunt. We find that cooperation and coordination can be achieved through the self-organization of the social network into strong and stable clusters of identical strategies.

1 Introduction and Previous Work

Evolutionary game theory has been traditionally applied to very large populations in which pairs of agents are drawn uniformly at random to play a given two-person one-shot game [1]. The population dynamics is such that those strategies that do better than average increase their share in the population, while those that do worse decline. The rest points of the dynamics are the equilibrium states, some of which are called *evolutionarily stable strategies* (ESS) and roughly correspond to the Nash equilibria (NE) of the game, i.e. those ensembles of strategies, one for each player, such that each strategy is a best response to the strategy of the other players [1]. This framework has allowed to satisfactorily explain a number of aspects and behavior in human and animal societies. However, there exist games in which either the equilibrium posited by the theory is logically and socially unsatisfying, or there is more than one equilibrium and no way to rationally choose between them, although some equilibrium clearly appears to be socially more efficient. The first kind of difficulty is illustrated by the well known *Prisoner's Dilemma* (PD), while the second situation is found, among others, in another paradigmatic game: the *Stag-Hunt*. These two representative games (see, for instance, [2,3] for more details) are two-person, two-strategies, symmetric games with the following payoff bi-matrix:

	C	D
C	(R,R)	(S,T)
D	(T,S)	(P,P)

R stands for the *reward* the two players receive if they both cooperate (C), P is the *punishment* for bilateral defection (D), and T is the *temptation*, i.e. the payoff that a player receives if it defects, while the other cooperates. In this case, the cooperator gets the *sucker's payoff* S. In both games, the condition $2R > T + S$ is imposed so that mutual cooperation is preferred over an equal probability of unilateral cooperation and defection. For the PD, the payoff values are ordered as: $T > R > P > S$. Defection is always the best rational individual choice in the PD; (D,D) is the unique *Nash equilibrium* (NE) and also an *evolutionarily stable strategy* (ESS) [1]. Mutual cooperation would be preferable but it is a strongly dominated strategy.

In the SH, the ordering is $R > T > P > S$, which means that mutual cooperation (C,C) is the best outcome, Pareto-superior, and a Nash equilibrium. However, there is a second equilibrium in which both players defect (D,D) and which is somewhat "inferior" to the previous one, although perfectly equivalent from a NE point of view. The (D,D) equilibrium is less satisfactory yet "risk-dominant" since playing it "safe" by choosing strategy D guarantees at least a payoff of P, while playing C might expose a player to a D response by her opponent, with the ensuing minimum payoff S. Here the dilemma is represented by the fact that the socially preferable coordinated equilibrium (C,C) might be missed for "fear" that the other player will play D instead. Although the PD has received much more attention in the literature than the SH, the latter is also very useful, especially as a metaphor of coordinate social behavior for mutual benefit [3].

In practice however, cooperation and coordination on common objectives is often seen in human and animal societies [2,3]. Coordinate behavior, such as having both players cooperating in the SH, is a bit less problematic as this outcome, being a Nash equilibrium, is not ruled out by theory. For the PD several mechanisms have been invoked to explain the emergence of cooperative behavior, such as repeated interaction, reputation, and belonging to a recognizable group [2]. Yet, Nowak and May [4] showed that when players are arranged according to a spatial structure and only interact with neighbors, a certain amount of cooperation can be sustained even when the game is played anonymously and without repetition. Nowak and May's study and much of the following work were based on regular structures such as two-dimensional grids. Nevertheless, we now know that actual social networks have a topological structure that is neither regular nor random. Instead, individuals may have a widely different number of neighbors [5]).

Some work has been done in recent years in the direction of using those more realistic kind of networks. In particular we mention work on scale-free networks [6], on Watts–Strogatz small-world graphs [7], and on model and real social networks [8]. These investigations have shown that a realistic structure of the society, with interactions limited to neighbors in the network, is sufficient for cooperative and coordinate behavior to emerge without making any particular assumption about the rationality of the actors. However, all the above mentioned studies have assumed a fixed population size and structure. But real social networks, such as friendship or collaboration networks, are not in an equilibrium state, they are open systems that continually evolve with new agents joining or leaving the network, and relationships being made or dismissed by agents already in the network. In the present work we re-introduce these coupled dynamics and we investigate under which conditions cooperative and coordinate behavior may emerge and be stable. We also study the topological structures of the resulting networks and their relationships with the strategic choices of the agents. Some previous work has been done on evolutionary games on dynamic networks [9,10,11]. The present study differs from those in the way in which links between agents are represented and interpreted, we study the agents' and link dynamics with a different stochastic rule, and we use a novel asynchronous update sequence.

This article is structured as follows. In sect. 2, we present our model of co-evolving dynamical network. In sect. 3, we discuss the simulation results and their significance for the social networks. We give our conclusions in sect. 4.

2 Model Description

Our model is strictly local. A player only uses information about the strength of the links with her first neighbors and the knowledge of her own payoff plus the strategies and indirectly the payoffs of her immediate neighbors. The model is an evolutionary one: players just adapt their behavior to imitate more successful strategies in their neighborhood with higher probability. In addition, they are able to locally assess the worth of an interaction and possibly dismiss a relationship that does not pay off enough. The model and its dynamics are described in detail in the following sections.

Network and Interaction Structure. The network of agents is represented as an undirected graph $G(V, E)$, where the set of vertices V represents the agents, while the set of edges (or links) E represents their symmetric interactions. The population size N is the cardinality of V. A neighbor of an agent i is any other agent j such that there is an edge $\{ij\} \in E$. The set of neighbors of i is called V_i and its cardinality is the degree k_i of vertex $i \in V$. The average degree of the network will be called $\bar{k}$.

Although there is formally a single undirected link between a player i and another player $j \in V_i$, we shall maintain two links: one going from i to j and another one in the reverse direction. Each link has a weight or "force" f_{ij} (respectively f_{ji}). This weight, say f_{ij}, represents in an indirect way the "trust" player i places in player j. It may take any value in $[0, 1]$ and its variation is dictated by the payoff earned by i in each encounter with j, as explained below. The link strengths can be seen as a kind of "memory" of previous encounters. However, they must be distinguished from the memory used in iterated games, in which players "remember" a certain number of previous moves and can thus conform their future strategy on the analysis of those past encounters [2]. Our interactions are strictly one-shot, i.e. players "forget" the results of previous rounds and cannot recognize previous partners and their possible playing patterns. We define a quantity s_i called *satisfaction* of an agent i as the mean weight of i's links: $s_i = \frac{\sum_{j \in V_i} f_{ij}}{k_i}$. Clearly $0 \leq s_i \leq 1$.

Initialization. The constant size of the network during the simulations is $N = 1000$. The initial graph is generated randomly with a mean degree $\bar{k} = 10$. This value of $\bar{k}$ is of the order of those actually found in many social networks (see, for instance, [5]). Players are distributed uniformly at random over the graph vertices with 50% cooperators. Forces between any pair of neighboring players are initialized at 0.5. There is another parameter q that has to be set; q is a real number in $[0, 1]$ and it represents the frequency with which an agent wishes to dismiss a link with one of its neighbors. The higher q, the faster the link reorganization in the network. It is an important consideration, as social networks may structurally evolve at widely different speeds, depending on the kind of interaction between agents. Agents are updated partially asynchronously, i.e. in each time step, a fraction of randomly chosen agents is updated simultaneously.

Strategy and Link Dynamics. When a given individual i is chosen to be activated it goes through the following steps:

- if the degree of agent i, $k_i = 0$ then player i is an isolated node. In this case a link with strength 0.5 is created from i to a player j chosen uniformly at random among the other $N - 1$ players in the network.
- otherwise,
  - either agent i updates its strategy according to a local *replicator dynamics* rule with probability $1 - q$ or, with probability q, agent i may delete a link with a given neighbor j and creates a new 0.5 force link with another node k ;
  - the forces between i and its neighbors V_i are updated

Let us now describe each step in more detail.

Strategy Evolution. We use a local version of replicator dynamics (RD) as described in [8]. The local dynamics of a player i only depends on its own strategy and on the strategies of the k_i players in its neighborhood V_i. Let π_{ij} be the payoff player i receives when interacting with neighbor j. The quantity $\widehat{\Pi}_i(t) = \sum_{j \in V_i} \pi_{ij}(t)$ is the *accumulated payoff* collected by player i at time step t. The rule according to which agents update their strategies is the conventional RD in which strategies that do better than the average increase their share in the population, while those that fare worse than average decrease. To update the strategy of player i, another player j is drawn at random from the neighborhood V_i. The probability of switching strategy is a monotonic increasing function ϕ of the payoff difference (here linear) [1]. Strategy s_i is replaced by s_j with probability

$$p_i = \phi(\widehat{\Pi}_j - \widehat{\Pi}_i). \tag{1}$$

Link Evolution. The active agent i, which has $k_i \neq 0$ neighbors will, with probability q, attempt to dismiss an interaction with one of its neighbors. This is done as follows. Player i looks at its satisfaction s_i. The higher s_i, the more satisfied the player, since a high satisfaction is a consequence of successful strategic interactions with the neighbors. Thus, there should be a natural tendency to try to dismiss a link when s_i is low. This is simulated by drawing a uniform pseudo-random number $r \in [0, 1]$ and breaking a link when $r \geq s_i$. Assuming that the decision is taken to cut a link, which one, among the possible k_i, should be chosen? Our solution again relies on the strength of the relevant links. First a neighbor j is chosen with probability proportional to $1 - f_{ij}$, i.e. the stronger the link, the less likely it will be chosen. This intuitively corresponds to i's observation that it is preferable to dismiss an interaction with a neighbor j that has contributed little to i's payoff over several rounds of play. However, in our system dismissing a link is not free: j may "object" to the decision. The intuitive idea is that, in real social situations, it is seldom possible to take unilateral decisions; we represent this by a probability $1 - (f_{ij} + f_{ji})/2$ with which j may refuse to be cut away. In other words, the link is less likely to be deleted if j appreciates i, i.e. when f_{ji} is high.

Assuming that the $\{ij\}$ link is finally cut, how is a new link to be formed? The solution adopted here is inspired by the observation that, in social networks, links are usually created more easily between people who have a mutual acquaintance than those who do not. First, a neighbor k is chosen in $V_i \setminus \{j\}$ with probability proportional to f_{ik}, thus favoring neighbors i trusts. Next, k in turn chooses player l in his neighborhood V_k using the same principle, i.e. with probability proportional to f_{kl}. If i and l are not connected, a link $\{il\}$ is created, otherwise the process is repeated in V_l. Again, if the selected node, say m, is not connected to i, a new link $\{im\}$ is established. If this also fails, a new link between i and a randomly chosen node is created. In all cases the new link is initialized with a strength of 0.5 in both directions.

Updating the Link Strengths. Once the chosen agents have gone through their strategy or link update steps, the strengths of the links are updated accordingly:

$$f_{ij}(t+1) = f_{ij}(t) + \frac{\pi_{ij} - \bar{\pi}_{ij}}{k_i(\pi_{max} - \pi_{min})},$$

where π_{ij} is the payoff of i when interacting with j, $\bar{\pi}_{ij}$ is the payoff earned by i playing with j, if j were to play his other strategy, and π_{max} (π_{min}) is the maximal (minimal) possible payoff obtainable in a single interaction. This update is performed in both directions, i.e. both f_{ij} and f_{ji} are updated $\forall j \in V_i$.

3 Simulation Results

3.1 Simulation Parameters

For each game, we can explore the entire game space by limiting our study to the variation of only two parameters per game. In the case of the PD, we set $R = 1$ and $S = 0$, and vary $1 \leq T \leq 2$ and $0 \leq P \leq 1$. For the SH we fix $R = 1$ and $S = 0$ and vary $0 \leq T \leq 1$ and $0 \leq P \leq T$. In the PD case, P is bounded by $R = 1$ and $S = 0$ in order to respect the ordering of the payoffs ($T > R > P > S$) and T's upper bound is equal to 2 due to the $2R > T + S$ constraint. In the SH, setting $R = 1$ and $S = 0$ determines the range of T and P (since this time $R > T > P > S$). Note that for this game the only valid value pairs of (T, P) are those that satisfy the $T > P$ constraint.

As stated in sect. 2, we used networks of size $N = 1000$, randomly generated with an average degree $\bar{k} = 10$ and randomly initialized with 50% cooperators and 50% defectors. In all cases, the parameters are varied between their two bounds in steps of 0.1. For each set of values, we carry out 50 runs of at most 20000 steps each, using a fresh graph realization in each run. A run is stopped when all agents are using the same strategy, in order to be able to measure statistics for the population and for the structural parameters of the graphs. The system is considered to have reached a pseudo-equilibrium strategy state when the strategy of the agents (C or D) does not change over 150 further steps, which means 15×10^4 individual updates. We speak of pseudo-equilibria and not of true evolutionary equilibria because, as we shall see below, the system never quite reaches a totally stable state in the dynamical systems sense.

3.2 Emergence of Cooperation

Fig. 1 shows cooperation results for the PD in contour plot form. As observed in other structured populations, cooperation may thrive in a small but non-negligible part of the parameter space. Thus, the added degree of freedom represented by the possibility of refusing a partner and choosing a new one does indeed help to find players' arrangements that help cooperation. This finding is in line with the results of [10,11]. Furthermore, the fact that our artificial society model differs from the latter two in several important ways also shows that the result is a rather robust one. When considering the dependence on the fluidity parameter q, one sees in fig. 1 that the higher q, the higher the cooperation level. This was expected since being able to break ties more often clearly gives cooperators more possibilities for finding and keeping fellow cooperators to interact with. Compared with the level of cooperation observed in simulations in static

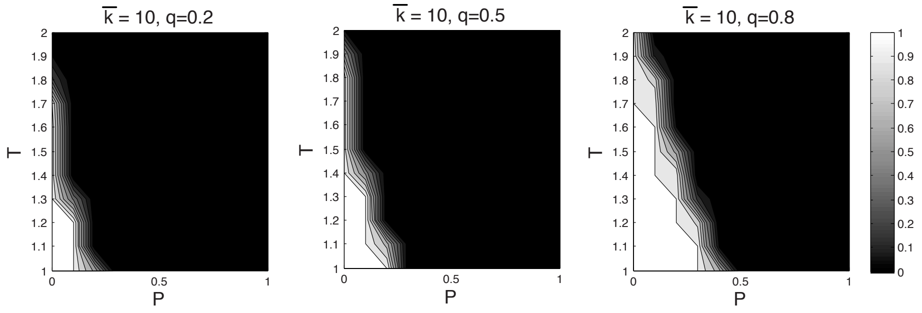


Fig. 1. Cooperation level for the PD in the game's configuration space. Darker gray means more defection.

networks, we can say that results are consistently better for co-evolving networks. For example, the typical cases with $\bar{k} = 10$ and $q = 0.5, 0.8$ show significantly more cooperation than what was found in model and real social networks in previous work [8]. Even when there is a much lower rewiring frequency, i.e. with $q = 0.2$, the cooperation levels are approximately as those observed in the mentioned study in which exactly the same replicator dynamics scheme was used to update the agents' strategies and the networks were of comparable size. The reason for this behavior is to be found in the added constraints imposed by the invariant network structure.

From the point of view of the evolutionary dynamics, it is interesting to point out that any given simulation run either ends up in full cooperation or full defection. When the full cooperation state of the population is attained, there is no way to switch back to defection by the intrinsic agent dynamics. In fact, all players are satisfied and have strong links with their cooperating neighbors. Even though a small amount of noise may still be present when deciding whether or not to rewire a link, since there are only cooperators around to imitate, there can be no strategy change and only very little link rewiring. On the other hand, well before this stable state is reached and there are still many defectors around, the system may experience some random drift that may drive it to full defection. The converse may also happen, but when the full defection state is reached, the situation is qualitatively different. In this case agents are unsatisfied, they will often try to rewire their links. However, all the other players around being also defectors, there will be constant changes of the local network structure. Thus the system will find itself in a fluctuating state, but this matters little for the bulk statistical properties of the population and of the network. To be assured that this is indeed the case, we have conducted some very long runs with all-defect end states. Global statistics do not change, except that the mean degree tends to increase slightly with time and the degree distribution function continues to evolve (see sect. 3.3).

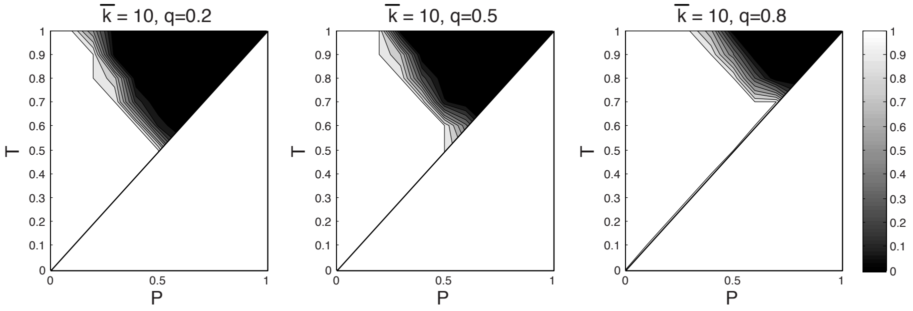


Fig. 2. Cooperation level for the SH game

Cooperation percentages as a function of the payoff matrix parameters for the SH game are shown in fig. 2 for $\bar{k} = 10$ and $q = 0.2, 0.5$, and 0.8. Note that in this case only the upper left triangle of the configuration space is meaningful. The SH is different from the PD since there are two evolutionarily stable strategies which are therefore also NEs: one population state in which everybody defects and the opposite one in which everybody cooperates (see sect. 1). Therefore, some runs will end up with all defect, while others will witness the emergence of full cooperation. In contrast, in the PD the only theoretically stable state is all-defect and cooperating states may emerge and be stable only by exploiting the graph structure and creating more favorable neighborhoods by breaking and forming ties. The value of the SH is in making manifest the tension that exists between the socially desirable state of full cooperation and the socially inferior but less risky state of defection [3]. The final outcome of a given simulation run depends on the size of the basin of attraction of either state, which is in turn a function of the relative values of the payoff matrix entries. To appreciate the usefulness of making and breaking ties in this game we can compare our results with what is prescribed by the standard RD solution. Referring to the payoff table of sect. 1, let's assume that the column player plays C with probability α and D with probability $1 - \alpha$. In this case, the expected payoffs of the row player are: $E_r[C] = \alpha R + (1 - \alpha)S$ and $E_r[D] = \alpha T + (1 - \alpha)P$. The row player is indifferent to the choice of α when $E_r[C] = E_r[D]$. Solving for α gives: $\alpha = (P - S)/(R - S - T + P)$. Since the game is symmetric, the result for the column player is the same and $(\alpha C, (1 - \alpha)D)$ is a NE in mixed strategies. Let us now use the following payoff values in order to bring them within the explored game space (NEs are invariant w.r.t. such a transformation [1]): $R = 1, T = 2/3, P = 1/3, S = 0$. Substituting in the previous expression gives $\alpha = 1/2$, i.e. the (unstable) polymorphic population should be composed by about half cooperators and half defectors. Now, if one looks at fig. 2 at the points where $P = 1/3$ and $T = 2/3$, one can see that this is approximately the case for the first image, within the limits of the approximations caused by the finite population size. On the other hand, in the middle image and, to a greater extent, in the rightmost image, this point in the game space corresponds to pure cooperation. In other words, the non-homogeneity of the network and an increased level of tie rewiring has allowed the cooperation basin to be enhanced with respect to the theoretical predictions of standard RD.

3.3 Structure of the Emerging Networks

Here we present a statistical analysis of the global and local properties of the networks that emerge when the pseudo-equilibrium states of the dynamics are attained. Let us first consider the evolution of the average degree $\bar{k}$. Although there is nothing in our model to prevent a change in the initial mean degree, the steady-state average connectivity tends to increase only slightly. For example, in the PD with $q = 0.8$ and $\bar{k}_{init} = 10$, the average steady-state (ss) value is $\bar{k}_{ss} \simeq 10.5$. Thus we see that, without imposing a constant $\bar{k}$, the latter nonetheless tends to increase only slightly, which nicely agrees with observations of real social networks.

The clustering coefficient C_i of a node i is defined as $C_i = 2E_i/k_i(k_i - 1)$, where E_i is the number of edges in the neighborhood of i. Thus C_i measures the amount of "cliquishness" of the neighborhood of node i and it characterizes the extent to which nodes adjacent to node i are connected to each other. The clustering coefficient of the graph is simply the average over all nodes: $C = \frac{1}{N}\sum_{i=1}^{N} C_i$ [5]. Random graphs are locally homogeneous and for them C is simply equal to the probability of having an edge between any pair of nodes independently. In contrast, real networks have local structures and thus higher values of C. Table 1 gives the average clustering coefficient $\bar{C} = \frac{1}{50}\sum_{i=1}^{50} C$ for three points in the game space which cover both cooperation and defection for both games (see figs. 1 and 2). It is apparent that the networks self-organize and acquire local structure, since the clustering coefficients are higher than those of the random graph with the same number of edges and nodes, which is $\bar{k}/N = 10/1000 = 0.01$. However, C is larger where cooperation predominates. The social networks develop local structures and the more so the higher the value of q.

Table 1. Clustering coefficient of the resulting networks. For both games we have an always cooperative case, a mixed case, and an alwasy defective case.

		$q = 0.2$	$q = 0.5$	$q = 0.8$
PD	$P = 1.1$ $T = 0.1$	0.1433	0.1937	0.2421
	$P = 1.8$ $T = 0.1$	0.0146	0.0176	0.2462
	$P = 1.1$ $T = 0.4$	0.0156	0.0404	0.0466
SH	$P = 0.9$ $T = 0.1$	0.1019	0.1564	0.2192
	$P = 0.9$ $T = 0.4$	0.0171	0.0254	0.2604
	$P = 0.9$ $T = 0.7$	0.0228	0.0363	0.0443

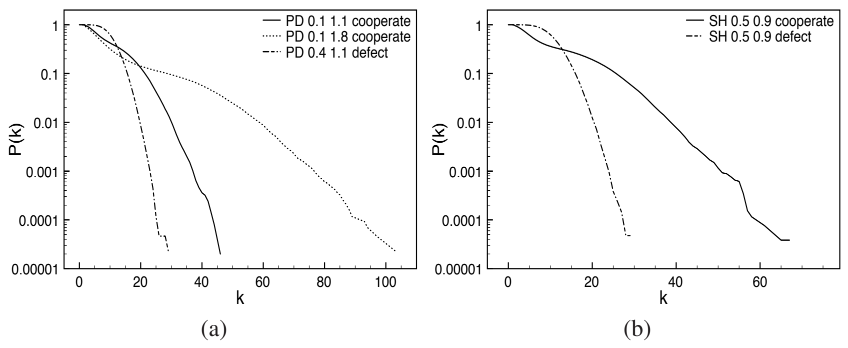


Fig. 3. Cumulative degree distributions. Average values over 50 runs. (a): PD, (b): SH. $q = 0.8$, $\bar{k} = 10$. Linear-log scales.

The *degree distribution function* (DDF) $p(k)$ of a of a graph represents the probability that a randomly chosen node has degree k [5]. Random graphs are characterized by DDF of Poissonian form, while social and technological real networks often show long tails to the right, i.e. there are nodes that have an unusually large number of neighbors [5]. The *cumulative degree distribution function* (CDDF) is just the probability that the degree is greater than or equal to k and has the advantage of being less noisy for high degrees. Fig. 3 (a) shows the CDDFs for the PD for three cases of which two are in the cooperative region and the third falls in the defecting region (see fig. 1). The dotted curve refers to a region of the configuration space in which there is cooperation in the average but it is more difficult to reach, as the temptation parameter is high (T=1.8, P=0.1). The curve has a rather long tail and is thus broad-scale in the sense that there is no typical degree for the agents. Therefore, in the corresponding network there are cooperators that are linked to many other cooperators. On the other hand, if one considers the dotted-dashed curve, which corresponds to a defecting region (T=1.1, P=0.4), it is clear that the distribution is much closer to normal, with a well-defined typical value of the degree. Finally, the third thick curve, which corresponds to a region where cooperation is more easily attained (T=1.1, P=0.1), also shows a rather faster decay of the tail than the dotted line and a narrower scale for the degree. Nevertheless, it is right-skewed, indicating that the network is no longer a pure random graph. Since we use linear-log scales, the dotted curve has an approximatively exponential or slower decay, given that a pure exponential would appear as a straight line in the plot. The tail of the thick curve decays faster than an exponential, while the dashed-dotted curve decays even faster. Almost the same observations also apply to the SH case (fig. 3 (b)). These are typical behaviors. When cooperation is difficult to reach, agents must exploit link-redirection in order for cooperators to stick together in sufficient quantities and protect themselves from exploiting defectors during the co-evolution. When the situation is either more favorable for cooperation, or defection easily prevails, network rearrangement is less radical. In the limit of long simulation times, the defection case leads to networks that have degree distribution close to Poissonian and are thus almost random.

4 Conclusions

By means of two well known games that represent conflicting decision situations we have studied the role of the dynamically networked society's structure in the establishment of global cooperative and coordinate behaviors, which are desirable outcomes for society's welfare. Starting from randomly connected players which only interact locally in a restricted neighborhood, and allowing agents to probabilistically and bilaterally dismiss unprofitable relations and create new ones, the stochastic dynamics lead to pseudo-equilibria of either cooperating or defecting agents. With respect to standard replicator dynamics results for mixing populations, we find that there is a sizable configuration space region in which cooperation may emerge and be stable for the PD, whereas the classical result predicts total defection. For the SH, where both all-cooperate and all-defect steady-states are theoretically possible, we show that the basin of attraction for cooperation is enhanced. The self-organizing mechanism consists in both games in forming dense clusters of cooperators which are more difficult to dissolve by exploiting defectors. While the beneficial effect of relational or geographical static population structures on cooperation was already known from previous studies, here we have shown that even more realistic dynamic social networks may allow cooperation to thrive.

Acknowledgments. E. Pestelacci and M. Tomassini are grateful to the Swiss National Science Foundation for financial support under contract number 200021-111816/1.

References

1. Weibull, J.W.: Evolutionary Game Theory. MIT Press, Boston (1995)
2. Axelrod, R.: The Evolution of Cooperation. Basic Books, Inc., New York (1984)
3. Skyrms, B.: The Stag Hunt and the Evolution of Social Structure. Cambridge University Press, Cambridge (2004)
4. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature 359, 826–829 (1992)
5. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
6. Santos, F.C., Pacheco, J.M.: Scale-free networks provide a unifying framework for the emergence of cooperation. Phys. Rev. Lett. 95, 098104 (2005)
7. Tomassini, M., Luthi, L., Giacobini, M.: Hawks and doves on small-world networks. Phys. Rev. E 73, 016132 (2006)
8. Luthi, L., Pestelacci, E., Tomassini, M.: Cooperation and community structure in social networks. Physica A 387, 955–966 (2008)
9. Luthi, L., Giacobini, M., Tomassini, M.: A minimal information prisoner's dilemma on evolving networks. In: Rocha, L.M. (ed.) Artificial Life X, pp. 438–444. MIT Press, Cambridge (2006)
10. Santos, F.C., Pacheco, J.M., Lenaerts, T.: Cooperation prevails when individuals adjust their social ties. PLOS Comp. Biol. 2, 1284–1291 (2006)
11. Zimmermann, M.G., Eguíluz, V.M.: Cooperation, social networks, and the emergence of leadership in a prisoner's dilemma with adaptive local interactions. Phys. Rev. E 72, 056118 (2005)

Preventing Premature Convergence in a Simple EDA Via Global Step Size Setting

Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Technická 2, 166 27 Prague 6, Czech Republic
posik@labe.felk.cvut.cz

Abstract. When a simple real-valued estimation of distribution algorithm (EDA) with Gaussian model and maximum likelihood estimation of parameters is used, it converges prematurely even on the slope of the fitness function. The simplest way of preventing premature convergence by multiplying the variance estimate by a constant factor k each generation is studied. Recent works have shown that when increasing the dimensionality of the search space, such an algorithm becomes very quickly unable to traverse the slope and focus to the optimum at the same time. In this paper it is shown that when isotropic distributions with Gaussian or Cauchy distributed norms are used, the simple constant setting of k is able to ensure a reasonable behaviour of the EDA on the slope and in the valley of the fitness function at the same time.

1 Introduction

Estimation of distribution algorithms (EDAs) [1] are a class of evolutionary algorithms (EAs) that do not use the crossover and mutation operators to create the offspring population. Instead, they build a probabilistic model describing the distribution of promising individuals and create offspring by sampling from the model. In real-valued spaces, such an algorithm can have a very simple structure which is depicted in Fig. 1.

If the Gaussian distribution is employed as the model of promising individuals ([2], [3], [4], [5]), and the parameters of the distribution, μ and σ, are learned by maximum likelihood (ML) estimation, the algorithm is very prone to premature convergence (i.e. the population converges on the slope of the fitness function) as recognized by many authors (see e.g. [3], [6], [7]). In [8], it was shown also theoretically that the distance traversed by a simple Gaussian EDA with truncation selection is bounded, and [9] showed similar results for tournament selection.

Many techniques that fight the premature convergence were developed, usually by means of artificially enlarging the ML estimate of variance of the learned distribution. In [6] it is suggested to use standard deviation greater than 1 when sampling the Gaussian distribution (e.g. to use $\mathcal{G}(0, 1.5)$). Adaptive variance scaling (AVS), i.e. enlarging the variance when better solutions were found and shrinking the variance in case of no improvement, was used along with various

```
1. Initialize the parameters $\mu^0 = (\mu^0_1, \ldots, \mu^0_D)$ and $\sigma^0 = (\sigma^0_1, \ldots, \sigma^0_D)$, $D$ is the
   dimensionality of the search space. Generation counter $t = 0$.
2. Sample $N$ offspring from the search distribution (use $\mu^t$ as the distribution
   center and $\sigma^t$ as relative scaling factors of individual components).
3. Evaluate the individuals.
4. Select the $\tau N$ best solutions (truncation selection).
5. Update parameters $\mu^{t+1}$ and $\sigma^{t+1}$ using the selected individuals.
6. Enlarge the $\sigma^{t+1}$ by a constant factor $k$ (global step size).
7. Advance generation counter: $t = t + 1$.
8. If termination condition is not met, go to step 2.
```

Fig. 1. Simple EDA analysed in this article

techniques to trigger the AVS only on the slope of the fitness function in [10] and [11]. The algorithm in Fig. 1, that suggests enlarging the population variance by a constant factor each generation, was studied in [12] where the minimal values of the 'amplification coefficient' were determined by a search in 1D case. In [13], the theoretical model of the algorithm behavior in 1D was used to derive the minimal and maximal admissible values for k. However, in [14] it was shown experimentaly that a constant multiplier does not ensure the desired properties of the algorithm when increasing the dimensionality of the search space.

In this article it is shown that when a modified Gaussian or Cauchy distribution is used instead of the standard Gaussian distribution, the simple approach with multiplying the population variance by a constant factor ensures the desired algorithm properties. Sec. 2 introduces the requirements constituting the bounds for a reasonable behaviour of the algorithm. Sec. 3 contains description of the probability distributions compared in this article. The results of the empirical study can be found in Sec. 4 and Sec. 5 concludes the paper.

2 Fundamental Requirements on EDA

According to [15], the optimal behaviour of the self-adaptive EAs in real spaces arises from balancing two antagonistic forces: (1) the variance shrinking effect of selection, and (2) the variance enlarging effect of the variational operators (distribution sampling, in our case). In this article, an approach of [14] is used where the combined effect of the selection and variation is taken into account.

Two simple fitness landscapes are used: a linear and a sphere function:

$$f_{\text{lin}}(\mathbf{x}) = x_1 \tag{1}$$

$$f_{\text{sphere}}(\mathbf{x}) = \sum_{d=1}^{D} x_d^2 \tag{2}$$

These functions can be regarded [15] as local approximations of the real fitness functions; the fitness landscape is often modelled as consisting of slopes and valleys (see e.g. [10], [16], [12]). The slopes and valleys are modelled with the linear (Eq. 1) and the sphere function (Eq. 2), respectively.

There are two fundamental requirements on the development of the population variance that ensure a reasonable behavior of the algorithm as a whole:

1. The variance *must not shrink on the slope.* This ensures that the population position is not bounded and that it eventually finds at least a local optimum.
2. The variance *must shrink in the valley.* In the neighborhood of the optimum, the algorithm must be allowed to converge to find the optimum precisely.

These two conditions constitute the bounds for the variance scaling factor k which must be large enough to traverse the slopes, but must not be too large to be able to focus to the optimum.

2.1 Bounds for *k*

The evolution of the model variance from one generation to another can be described as follows: (1) sample new individuals with variance $(\sigma^t)^2$, (2) select the best individuals, and (3) compute the variance $(\sigma^{t+1})^2$ for the next sampling. Without selection and using ML estimate, the two variances are expected to be the same. For our two fitness landscapes, the selection reduces the variance, thus

$$(\sigma^{t+1})^2 = (\sigma^t)^2 \cdot c, \tag{3}$$

where c is the ratio of the population variances in two consecutive generations, t and $t+1$, and $c < 1$ in our case. Of course, the ratio c differs for various fitness landscapes, thus it will be designated as c_{slope} and c_{valley}, respectively.

As already said in the introduction, the simplest method of preventing premature convergence is to enlarge the estimated standard deviation σ by a constant factor k (step 6 of the algorithm in Fig. 1). Thus

$$\sigma^{t+1} = k \cdot \sigma^t \cdot \sqrt{c} \tag{4}$$

In order to prevent the premature convergence on the slope, the ratio of the consecutive standard deviations should be at least 1, i.e.

$$\frac{\sigma^{t+1}}{\sigma^t} = k \cdot \sqrt{c_{\text{slope}}} \geq 1, \text{ thus } k \geq \frac{1}{\sqrt{c_{\text{slope}}}} \overset{\text{def}}{=} k_{\min}. \tag{5}$$

On the other hand, to be able to focus to the optimum, the model must be allowed to converge in the valley. The ratio of the two consecutive standard deviations should be lower than 1, i.e.

$$\frac{\sigma^{t+1}}{\sigma^t} = k \cdot \sqrt{c_{\text{valley}}} < 1, \text{ thus } k < \frac{1}{\sqrt{c_{\text{valley}}}} \overset{\text{def}}{=} k_{\max} \tag{6}$$

Joining these two conditions together gives us the bounds for the constant k:

$$k_{\min} = \frac{1}{\sqrt{c_{\text{slope}}}} \leq k < \frac{1}{\sqrt{c_{\text{valley}}}} = k_{\max} \tag{7}$$

In this paper, the value of k is called *admissible* if it satisfies condition 7.

3 Probability Distributions

Although [13] theoretically deduced bounds for k in case of 1D Gaussian distribution, in [14] it was shown that the process sketched above does not work with increasing dimensionality, since the interval of admissible k diminishes and eventually vanishes. This is due to the fact that the variance after selection in the neighborhood of the valley (sphere function) increases with dimensionality, thus $k_{\max}$ must be successively smaller and eventually gets lower than $k_{\min}$.

In this article, a distribution that does not exhibit this unpleasant behaviour is sought for. Three distributions are compared.

Standard Gaussian distribution (designated as $\mathcal{G}$)**.** Probably the most often used distribution in real-valued evolutionary algorithms. The 1D normal distribution with zero mean and variance σ^2 has the following p.d.f.:

$$f_{\mathcal{N}(0,\sigma^2)}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \tag{8}$$

Sampling process. D-dimensional realizations of the standard normal distribution can be created by sampling each component independently from the 1D standard normal distribution.

Isotropic distribution with 1D Gaussian norm (designated as $\mathcal{G}^{\text{iso}}$)**.**[1] Used in the hope that it preserves some features of the 1D Gaussian distribution.

Sampling process. 1D version of $\mathcal{G}^{\text{iso}}$ is the same as 1D version $\mathcal{G}$. The multidimensional versions of $\mathcal{G}^{\text{iso}}$ can be created by (1) sampling the direction vector uniformly on the unit hypersphere[2], and (2) by multiplying the vector by a factor sampled from χ-distribution with 1 degree of freedeom. The χ-distribution describes norms of vectors generated from $\mathcal{G}$.

Isotropic distribution with 1D Cauchy norm (designated as $\mathcal{C}^{\text{iso}}$)**.** Selected for the comparison to show the effects of heavy tails (if any). The 1D Cauchy distribution with median 0 and upper quartile γ has the following p.d.f.:

$$f_{\mathcal{C}(0,\gamma)} = \frac{1}{\pi} \frac{\gamma}{x^2 + \gamma^2} \tag{9}$$

Standard Cauchy distributed values with $\gamma = 1$ can be be obtained by sampling two values from $\mathcal{G}$ and dividing them.

Sampling process. D-dimensional realizations of $\mathcal{C}^{\text{iso}}$ can be sampled similarly as $\mathcal{G}^{\text{iso}}$ with the exception that the multiplication factor is sampled from 1D Cauchy distribution instead of 1D Gaussian.

[1] In this article, the term *isotropic* is not meant as a feature of the distribution (standard normal distribution is isotropic as well); it describes the sampling process.

[2] Sampling a vector on a unit hypersphere can be achieved e.g. by sampling D-dimensional standard normal distribution and dividing the resulting vector by its norm.

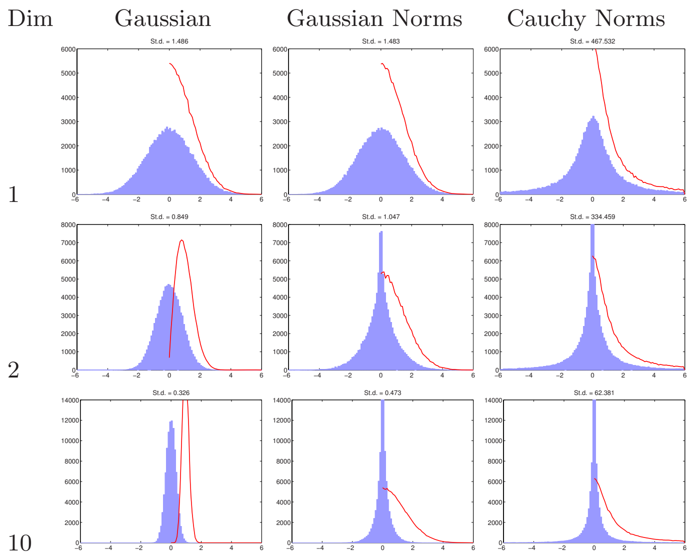


Fig. 2. The distribution of the first coordinate (the histograms) and the distribution of the vector norms (solid line) for $\mathcal{G}$, $\mathcal{G}^{\text{iso}}$, and $\mathcal{C}^{\text{iso}}$, and for the search space dimensions 1, 2, and 10

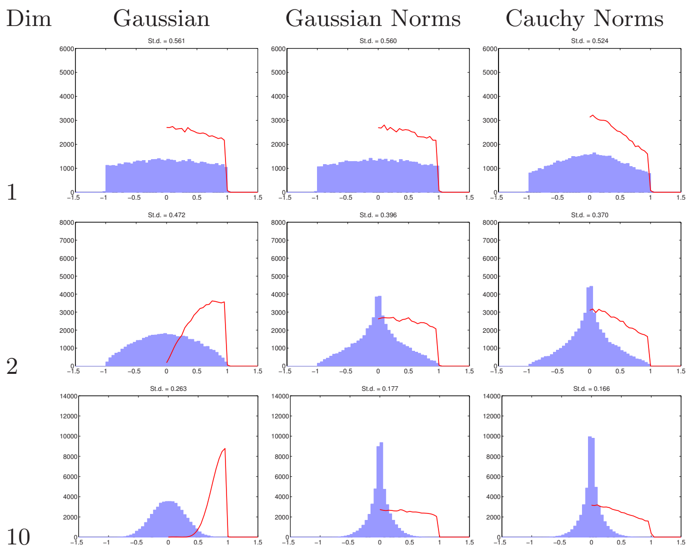


Fig. 3. After selection with sphere function. The distribution of the first coordinate (the histograms) and the distribution of the vector norms (solid line) for $\mathcal{G}$, $\mathcal{G}^{\text{iso}}$, and $\mathcal{C}^{\text{iso}}$, and for the search space dimensions 1, 2, and 10. Note that the distribution of vector norms (solid line) is cut off at $x = 1$ due to the modification of sampling process described in Sec. 3.1.

These distributions were already studied in several works from different points of view. In [17], the local convergence rates of evolutionary algorithms with Gaussian and Cauchy mutations are estimated and compared. In [18], the convergence to a local optimum was studied as well, along with the ability to locate narrow valleys and the influence of the dimensionality on the exploration efficiency. The usefulness of the Cauchy distributions in case of multimodal optimization was explored in [19]. In this article it is studied if these distributions allow for the simple constant setting of the global step size.

3.1 Modification of Vector Norms

It was deliberately decided to normalize[3] the vector norms of all three distributions in such a way, that the 100τ-percentile of the distribution of norms equals to 1. This is achieved simply by

– dividing the $\mathcal{G}$-distributed vectors by the value of inverse cumulative distribution function (i.c.d.f) of the χ distribution with D degrees of freedom (d.o.f.) at point τ, i.e. $\mathbf{x}_m = \mathbf{x}/CDF^{-1}_{\chi_D}(\tau)$, $\mathbf{x} \sim \mathcal{G}$,
– dividing the $\mathcal{G}^{\text{iso}}$-distributed vectors by the value of the i.c.d.f. of the χ distribution with 1 d.o.f. at point τ, i.e. $\mathbf{x}_m = \mathbf{x}/CDF^{-1}_{\chi_1}(\tau)$, $\mathbf{x} \sim \mathcal{G}^{\text{iso}}$, or by
– dividing the $\mathcal{C}^{\text{iso}}$-distributed vectors by the value of the i.c.d.f. of the standard Cauchy distribution at point $(1+\tau)/2$, i.e. $\mathbf{x}_m = \mathbf{x}/CDF^{-1}_{\mathcal{C}}(\frac{1+\tau}{2})$, $\mathbf{x} \sim \mathcal{C}^{\text{iso}}$, respectively.

The distributions of sampled data points and their norms are depicted in Fig. 2. The fact that the 100τ-percentile of the norm distribution is equal to 1 is demonstrated in Fig. 3 which shows the distributions of selected data points when sphere function is used. The frequency of norms of the selected data points is cut off at value 1.

4 Experiments, Results and Discussion

The bounds for k for all three distributions were found experimentaly. The lower bound $k_{\min}$ is found by using the f_{lin}, the upper bound $k_{\max}$ is found by experiments with f_{sphere}. During the experiments, the value of standard deviation of coordinate x_1 is tracked and it is checked if it increases or decreases (on average). The bisection method is used to determine the value of k for which the variance stays the same (with certain tolerance).

The population size 1,000 was used in all experiments. To determine each particular $k_{\min}$ (and $k_{\max}$), 10 independent runs of 100 generations were carried out. Each run was started with initial parameters $\mu^0 = 0$ and $\sigma^0 = 1$ ensuring that the processes are started in the stationary state. During each run, the standard deviation of x_1 was tracked; this gives 10 values of st.d. for each of 100

[3] There is no special need for the normalization. With the normalization, however, the graphs in Figs. 4 and 5 show more regular patterns and are more comparable.

generations. To this data, a linear function of the form $E(\log(\text{st.d.})) = a \cdot \text{gen} + b$ was fitted ('gen' is the generation counter) using simple linear regression which should be adequate type of model. The sign of the learned parameter a was used to decide, if the variances increase or decrease during the run.

The bounds of k found for $\mathcal{G}$ can be seen in Fig. 4. For 1D search space there exists an interval of admissible values of k for all tested selection proportions τ. However, with increasing dimensionality, the value of $k_{\min}$ grows faster than $k_{\max}$ for all values of τ, and for dimensions greater then 5 there is no admissible k (which would ensure effective traversing of slopes and focusing to the optimum in the same time). This is in accordance with the results in [13] and [14].

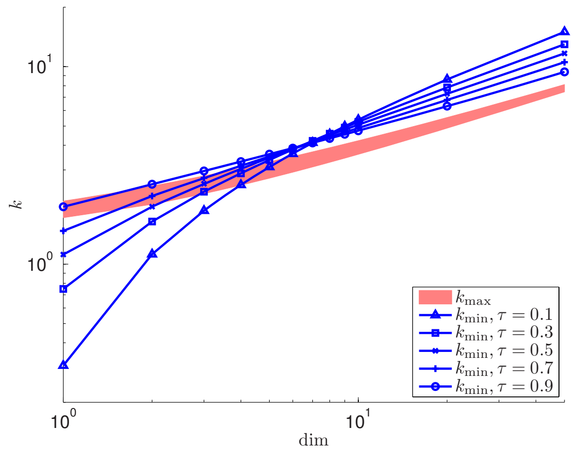


Fig. 4. Minimal and maximal values of k for the Gaussian distribution. (The lines for the respective $k_{\max}$ are very close to each other; without losing the big picture they were replaced by the shaded region.) It can be observed that for $D > 5$ the $k_{\min}$ is greater than $k_{\max}$ for all tested selection proportions τ and the admissible interval for k does not exist!

The same figures when $\mathcal{G}^{\text{iso}}$ is used are depicted in Fig. 5, left. The results are completely different now! For all but the highest values of τ, there seems to exist an interval of admisible values of k and this interval does not shrink with incresing dimensionality.

The situation for $\mathcal{C}^{\text{iso}}$ distribution is even better, see Fig. 5, right. The size of admissible interval for k does not shrink so much when increasing the selection proportion τ, as was the case for $\mathcal{G}^{\text{iso}}$.

It can be also observed that for the isotropic distributions and a particular value of selection proportion τ, the ratio $k_{\max}/k_{\min}$ stays almost the same regardless of the dimensionality. This observation could be used to create a simple equation for the setting of k in relation to τ and the dimensionality. Of course, optimal setting of k depends on the problem, on the initial values of μ^0 and σ^0, and can also depend on the search distribution used. At this moment, it is not

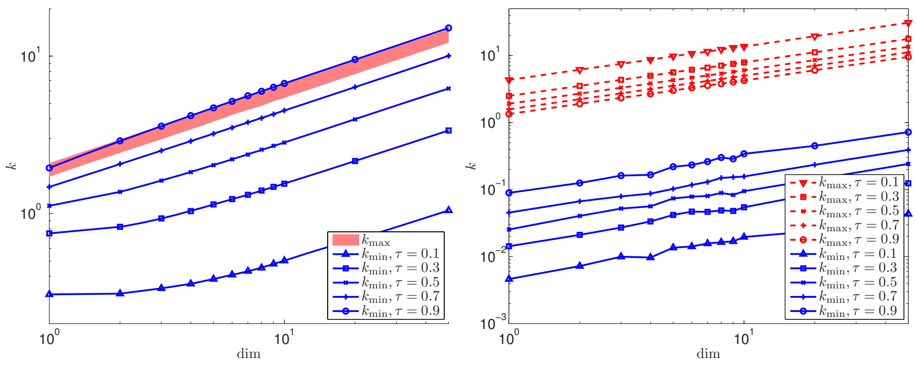


Fig. 5. Minimal and maximal values of k for the isotropic Gaussian (on the left) and isotropic Cauchy (on the right) distributions. (The lines for the respective $k_{\max}$ of the $\mathcal{G}^{\text{iso}}$ distribution are very close to each other; without losing the big picture they were replaced by the shaded region.) It can be observed that the admissible interval for k exists and does not shrink with the dimensionality for almost all tested selection proportions τ and for both tested isotropic distributions.

clear if it is better "on average" to set k only slightly above $k_{\min}$, slightly below $k_{\max}$, or somewhere in the middle.

As already said in the introduction, in [6] the authors showed that their EDA with truncation selection with $\tau = 0.3$ which used the value of 1.5 for the standard deviation of the Gaussian distribution was able to find the optimum of the 10D Rosenbrock function while EDA without this modification (using ML estimate of σ) converged prematurely. The value 1.5 can be transformed to the context of this article; the corresponding $k = 1.5 \cdot CDF^{-1}_{\chi_{10}}(0.3) \approx 4$. Looking at the Fig. 4 (dim=10, $\tau = 0.3$) we can see that this value is not admissible; it lies somewhere in the shaded region of $k_{\max}$, below $k_{\min}$. Thus, the population variance was shrinking during the whole evolution (as shown in [6]). The shrinking was a bit slower, however, than when using ML estimate of σ giving the algorithm the time needed to find the global optimum. The algorithm was started from the origin. If it was started from a more distant point, the results obtained in this article suggest that the optimum would not be reached.

The adaptive variance scaling approach (AVS) presented in [10] and [11] should work even for the isotropic distributions used in this article. Since it is a dynamic scheme for setting the k, it needn't be limited to admissible values of k. For the algorithm it is often profitable to set $k > k_{\max}$ when on slope, or to set $k < k_{\min}$ when in the valley which ensures faster traversal of slopes and faster convergence to the optimum, respectively. On the other hand, AVS alone is an iterative update scheme and it can take several generations to switch the scaling from slope-style to valley-style or vice versa. That is the reason behind the triggers introduced in [10] and [11] which should decide if the population is on the slope or in the valley and trigger the AVS only on the slope; in the valley, the ML estimate of σ is used without scaling. The right behavior of such an algorithm is largely determined by the ability of the trigger to decide correctly whether to trigger the scaling. The results of this article can thus be useful for these algorithms in two ways: (1) if the trigger is good, the scaling factor can be set to at least $k_{\min}$ on the slope, and at most to $k_{\max}$ in the valley, or (2) if the trigger makes mistakes, the algorithm can use the admissible interval of $\langle k_{\min}, k_{\max})$ as a safeguard.

5 Summary and Future Work

This article aimed at simple way of preventing premature convergence of a simple EDA. The variance of the distribution estimated from the selected data is increased by the factor (or global step size) k each generation, artificially keeping the sufficient diversity in the population.

Recent works have shown that when Gaussian distribution is used, a constant value of k which would ensure a reasonable behaviour of the algorithm on the slopes *and* in the valleys of the fitness function exists only for low-dimensional spaces.

The situation is much better when isotropic distribution with Gaussian or Cauchy norms is used. Both of these two distributions ensure the existence of the admissible interval for k for a broad range of selection proportions τ and search space dimensionalities. Moreover, the ratio $k_{\max}/k_{\min}$ stays almost the same for the isotropic distributions, with Cauchy distribution giving larger margin.

Compiling a practically appliable heuristic for setting the value of k, building a real working optimization algorithm based on these principles, and its comparison with other scaling techniques remain as the future work. It would be also appealing to explore this technique in combination with other selection schemes different from the truncation selection.

Acknowledgements

The project was supported by the Ministry of Education, Youth and Sport of the Czech Republic with the grant No. MSM6840770012 entitled “Transdisciplinary Research in Biomedical Engineering II”.

References

1. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. GENA. Kluwer Academic Publishers, Dordrecht (2002)
2. Larrañaga, P., Lozano, J.A., Bengoetxea, E.: Estimation of distribution algorithms based on multivariate normal distributions and gaussian networks. Technical Report KZZA-IK-1-01, Dept. of Computer Science and Artificial Intelligence, University of Basque Country (2001)
3. Bosman, P.A.N., Thierens, D.: Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In: PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, London, UK, pp. 767–776. Springer, Heidelberg (2000)
4. Rudlof, S., Köppen, M.: Stochastic hill climbing by vectors of normal distributions. In: First Online Workshop on Soft Computing, Nagoya, Japan (1996)

5. Ahn, C.W., Ramakrishna, R.S., Goldberg, D.E.: Real-coded bayesian optimization algorithm: Bringing the strength of BOA into the continuous world. In: Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E.K., Darwan, P.J., Dasgupta, D., Floreano, D., Foster, J.A., Harman, M., Holland, O., Lanzi, P.L., Spector, L., Tettamanzi, A., Thierens, D., Tyrrell, A.M. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 840–851. Springer, Heidelberg (2004)
6. Yuan, B., Gallagher, M.: On the importance of diversity maintenance in estimation of distribution algorithms. In: Beyer, H.G., O'Reilly, U.M. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2005, vol. 1, pp. 719–726. ACM Press, New York (2005)
7. Ocenasek, J., Kern, S., Hansen, N., Koumoutsakos, P.: A mixed bayesian optimization algorithm with variance adaptation. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 352–361. Springer, Heidelberg (2004)
8. Grahl, J., Minner, S., Rothlauf, F.: Behaviour of UMDAc with truncation selection on monotonous functions. In: IEEE Congress on Evolutionary Computation, CEC 2005, vol. 3, pp. 2553–2559 (2005)
9. Gonzales, C., Lozano, J., Larranaga, P.: Mathematical modelling of UMDAc algorithm with tournament selection. International Journal of Approximate Reasoning 31(3), 313–340 (2002)
10. Grahl, J., Bosman, P.A.N., Rothlauf, F.: The correlation-triggered adaptive variance scaling IDEA. In: Proceedings of the 8th annual conference on Genetic and Evolutionary Computation Conference - GECCO 2006, pp. 397–404. ACM Press, New York (2006)
11. Bosman, P.A.N., Grahl, J., Rothlauf, F.: SDR: A better trigger for adaptive variance scaling in normal EDAs. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation, pp. 492–499. ACM Press, New York (2007)
12. Yuan, B., Gallagher, M.: A mathematical modelling technique for the analysis of the dynamics of a simple continuous EDA. In: IEEE Congress on Evolutionary Computation, CEC 2006, Vancouver, Canada, pp. 1585–1591. IEEE Press, Los Alamitos (2006)
13. Pošík, P.: Gaussian EDA and truncation selection: Setting limits for sustainable progress. In: IEEE SMC International Conference on Distributed Human-Machine Systems, DHMS 2008, Athens, Greece. IEEE, Los Alamitos (2008)
14. Pošík, P.: Truncation selection and gaussian EDA: Bounds for sustainable progress in high-dimensional spaces. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., McCormack, J., O'Neill, M., Romero, J., Rothlauf, F., Squillero, G., Uyar, A.Ş., Yang, S. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 525–534. Springer, Heidelberg (2008)
15. Beyer, H.G., Deb, K.: On self-adaptive features in real-parameter evolutionary algorithms. IEEE Trans. on Evol. Comp. 5(3), 250–270 (2001)
16. Grahl, J., Bosman, P.A.N., Minner, S.: Convergence phases, variance trajectories, and runtime analysis of continuous EDAs. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 516–522. ACM Press, New York (2007)
17. Rudolph, G.: Local convergence rates of simple evolutionary algorithms with cauchy mutations. IEEE Transactions on Evolutionary Computation 1, 249–258 (1997)
18. Obuchowicz, A.: Multidimensional mutations in evolutionary algorithms based on real-valued representation. Int. J. Systems Science 34(7), 469–483 (2003)
19. Hansen, N., Gemperle, F., Auger, A., Koumoutsakos, P.: When do heavy-tail distributions help? In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 62–71. Springer, Heidelberg (2006)

A Steady-State Genetic Algorithm with Resampling for Noisy Inventory Control*

Steven Prestwich[1], S. Armagan Tarim[2], Roberto Rossi[1], and Brahim Hnich[3]

[1] Cork Constraint Computation Centre, University College, Cork, Ireland
s.prestwich@cs.ucc.ie, r.rossi@4c.ucc.ie
[2] Department of Management, Hacettepe University, Turkey
armagan.tarim@hacettepe.edu.tr
[3] Faculty of Computer Science, Izmir University of Economics, Turkey
brahim.hnich@ieu.edu.tr

Abstract. Noisy fitness functions occur in many practical applications of evolutionary computation. A standard technique for solving these problems is fitness resampling but this may be inefficient or need a large population, and combined with elitism it may overvalue chromosomes or reduce genetic diversity. We describe a simple new resampling technique called Greedy Average Sampling for steady-state genetic algorithms such as GENITOR. It requires an extra runtime parameter to be tuned, but does not need a large population or assumptions on noise distributions. In experiments on a well-known Inventory Control problem it performed a large number of samples on the best chromosomes yet only a small number on average, and was more effective than four other tested techniques.

1 Introduction

In many real-world applications of Genetic Algorithms (GAs) and other Evolutionary Computation algorithms, the fitness function is *noisy*: that is, the fitness of a chromosome cannot be computed directly but must be averaged over a number of samples. Examples include the learning of randomised games such as Backgammon, human-computer interaction, and simulation problems for which we wish to evolve a robust plan. The standard deviation of the sample mean of a random variable with standard deviation σ is $\sigma/\sqrt{n}$ where n is the number of samples, so a large number of samples may be needed for very noisy fitness functions.

Several techniques for handling fitness noise in EAs are surveyed in [4,13]: the use of sampling to obtain an average fitness reduces noise; increasing the population size makes it harder for an unfit chromosome to displace a fitter one

* S.A. Tarim and B. Hnich are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027. R. Rossi is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research (CTVR) and Grant No. 05/IN/I886.

(a point also made by [10]) and can be viewed as a form of implicit averaging; and *rescaled mutation* samples distant points in the search space then moves a small distance toward them. [5] propose regression to estimate the fitness of neighbouring chromosomes. [1] vary sample rates across both chromosomes and generations in a generational GA. [18] record fitness levels in a search history, and use a stochastic model of fitness levels to locate new points in the search space. [3] use a threshold selection heuristic for accepting chromosomes. [17] adapt the sampling rate to different regions of the search space, a technique they call *dynamic resampling*. [19] use a Bayesian approach to sampling called Optimal Computing Budget Allocation, which assumes normally distributed noise.

A popular approach is to use a *Noisy Genetic Algorithm* (NGA) which computes the fitness of each chromosome by averaging over a number of samples [9,11,14,15]. Following [1] we shall refer to this as *static sampling*, and refer to this algorithm as NGAs. NGAs wastes considerable time evaluating unpromising chromosomes, but it can be improved by linearly increasing the number of samples with search time, starting from a low value [21,27]. We shall refer to this as *incremental sampling* and the resulting algorithm as NGAi. However, though NGAs and NGAi have been used to solve real problems, they may not be the most efficient approach. It is pointed out in [22] that a reduction in noise is not necessary for *every* chromosome, only for the *best* ones. Of course, this entails discovering which are the best chromosomes without performing a large number of samples, but poor chromosomes might become apparent after just a few samples.

An alternative technique is to *resample* chromosome fitness: that is, some chromosomes are allowed to survive for more than one generation, and their fitness is periodically recomputed to refine the estimate. Various heuristics may be used to decide when to discard a chromosome. [22] experiments with averaging over a small number of samples, and guiding resampling by a statistical test which assumes Gaussian noise but is considered to be robust under non-Gaussian noise. [12] uses the standard deviation of the fitness to correct for its noise, again under assumptions on noise distribution. Resampling and the common heuristic of *elitism* do not always combine well. [6] show that, with an elitist GA, the probabilistic method of [12] is inferior to a resampling approach. [2] show that, in Evolutionary Strategies that allow fitness values to survive for more than one generation, failure to resample can lead to systematic overvaluation of chromosomes. [8] found that, when applying co-evolutionary learning to the noisy task of learning how to play Backgammon, more sampling can have a bad effect on the learning besides incurring overhead. It causes less fit chromosomes to be pruned more quickly which reduces genetic diversity too drastically, especially with small populations. Despite these drawbacks, resampling and elitism have been successfully combined. [25] describe an extension of the Simple (generational) GA that maintains a list of the fittest solutions found so far, while increasing the number of samples as search proceeds as in NGAi; they also increase the population size during search.

Another successful resampling elitist GA is the *Kalman-extended Genetic Algorithm* (KGA) [23], designed for problems whose fitness is both noisy and nonstationary. It adapts its sampling rate for each chromosome individually, based on techniques from Kalman filtering. Removing the nonstationary aspects of KGA yields a steady-state algorithm that evaluates the fitness of each new chromosome just once before adding it to the population, then replaces the least-fit population member by the new chromosome. Alternate iterations are devoted to resampling chromosomes that are already in the population. The current fitness estimate of a chromosome is the mean over all its samples. In KGA a chromosome is selected for resampling according to its current fitness estimate and how many times it has already been sampled (which is a measure of the fitness uncertainty): choose the chromosome with fewest samples, among those whose fitness estimates are greater than the population fitness mean minus the population fitness standard deviation. The intuition behind this approach is that unfit chromosomes with high fitness estimate based on only a few samples will be resampled, and their low fitness will become apparent. We shall refer to this as *Kalman sampling*.

In this paper we investigate resampling strategies for the steady-state (therefore elitist) GENITOR algorithm [26]. Our aim is to find a simple resampling strategy that can be used with a steady-state GA, does not assume any noise properties, does not require a large population, resamples fit chromosomes many times to avoid overvaluation, yet on average uses only a few samples per chromosome. We find it necessary to introduce a new runtime parameter that requires manual tuning, but this might be automated in future work. We demonstrate our technique on a well-known problem from Inventory Control. Section 2 describes our algorithm, Section 3 describes the problem we attempt to solve, Section 4 presents experimental results, and Section 5 concludes the paper.

2 The Algorithm

We use a single GA in our experiments: a basic version of GENITOR [26] without refinements such as a gene to determine crossover probability. GENITOR is a steady-state GA that, at each iteration, selects two parent chromosomes, breeds a single offspring by (optional) crossover followed by mutation, evaluates it, and uses it to replace the least-fit member of the population. We use random parent selection, and standard uniform crossover applied with a crossover probability 0.5: if it is not applied then a single parent is selected and mutated. In our problem (described below) each gene can take any of 100 integer values, plus a special value denoted by NULL. Because of the special nature of the NULL value we select it with probability 0.5, otherwise randomly select one of the 100 integer values. Mutation is applied to a chromosome once with probability 0.5, twice with probability 0.25, three times with probability 0.125, and so on. A small population of size 30 is used. We assume that at least U samples are required to obtain a sufficiently reliable fitness estimate, and in experiments we will use the large value $U = 1000$. Thus we face the challenge of sampling effectively without incurring the drawbacks described above: inefficiency, lack of genetic diversity, or overvaluation, while using only a small population.

This is our basic GA but we have yet to specify a sampling strategy to cope with fitness noise. We will compare five resampling strategies, three of which are well-known: static sampling (as in NGAs) in which we take U samples for each chromosome, incremental sampling (as in NGAs) in which we take a variable number of samples per chromosome that linearly increases from 1 to U during the GA execution, and Kalman sampling (as in KGA). The other two strategies are new.

Our first new strategy tries to combine the rapid convergence of Kalman sampling with the reliability of static sampling. It applies Kalman sampling but with a number $S \geq 1$ of samples to initialise and resample chromosomes, with the best value of S to be determined by experiment. We shall refer to this as *Kalman averaged sampling* and our GA with this sampling scheme as KASGA. It is inspired by a note in [1] stating that if the fitness variance in the population is small compared to the noise variance then a GA will make no progress, and it becomes necessary to increase the sample rate. It is also inspired by the use of a small number of samples for evolutionary algorithms in [22].

Our second new strategy also takes S samples each time a chromosome is selected for (re)sampling, but it resamples the chromosome with highest fitness, ignoring chromosomes that already have U samples. Note that if $S < U$ then there is always at least one chromosome with fewer than U samples: the most recently created chromosome, which only has S samples. Note also that we normally choose S to be a divisor of U to avoid unnecessary resampling, but this is not strictly required. We shall call this scheme *greedy averaged sampling* because it greedily resamples the most promising chromosome, based on current fitness estimates. Combining this with the GA we obtain a new algorithm we shall call the Greedy Average Sample GA (GASGA). This is our main contribution and it is summarised in Figure 1.

```
GASGA($S, P, U$)
  create population of size $P$
  evaluate population using $S$ samples
  while not(termination condition)
    select two parents
    breed one offspring O
    evaluate O using $S$ samples
    replace least-fit chromosome by O
    select fittest chromosome F with #samples $< U$
    re-evaluate F using $S$ samples
  output fittest chromosome
```

Fig. 1. GASGA pseudo-code

3 An Inventory Control Problem with Uncertainty

The problem we consider is as follows. Given a planning horizon of N periods and a demand for each period $t \in \{1, \dots, N\}$, which is a random variable with a given probability density function; we assume that these distributions are normal, though this is not required by our GA. Demands occur instantaneously at the beginning of each time period and are *non-stationary* (can vary from period to period), and demands in different periods are independent. A fixed delivery cost a is incurred for each order, a linear holding cost h is incurred for each product unit carried in stock from one period to the next, and a linear stockout cost s is incurred for each period in which the net inventory is negative (it is not possible to sell back excess items to the vendor at the end of a period). The aim is to find a replenishment plan that minimizes the expected total cost over the planning horizon.

Different inventory control policies can be adopted to cope with this and other problems. A policy states the rules used to decide when orders are to be placed and how to compute the replenishment lot-size for each order. (The term *policy* here refers to the *form* of the plan, whereas in some fields such as Artificial Intelligence a policy refers to an *actual* plan. We use the term in both senses, and the meaning should be clear from the context.) One possibility is the *replenishment cycle policy* (R, S) [20]. With non-stationary demands this policy takes the form (R^n, S^n) where R^n denotes the length of the n^{th} replenishment cycle and S^n the order-up-to-level for replenishment. In this policy a wait-and-see strategy is adopted, under which the actual order quantity for replenishment cycle n is determined only after the demand in former periods has been realized. The order quantity is computed as the amount of stock required to raise the closing inventory level of replenishment cycle $n-1$ up to level S^n. To provide a solution we must populate both the sets R^n and S^n for $n = \{1, \ldots, N\}$. The (R, S) policy yields plans of higher cost than optimal but has been formulated to reduce *nervousness* in inventory control, and is more often used in practice.

There are more efficient algorithms which are guaranteed to yield optimal policies (under reasonable simplifying assumptions) so a GA would not be applied to precisely this problem in practice. However, if we complicate the problem in simple but realistic ways, for example by adding order capacity constraints or dropping the assumption of independent demands, these efficient algorithms become unusable. In contrast, a GA can be used almost without modification. Thus the problem is useful as a representative of a family of more complex problems.

The replenishment cycle policy can be modelled as follows. Each chromosome represents a single policy, each gene corresponds to a period n, an allele specifies the order-up-to level or the lack of an order (denoted here by the special value NULL) for that period, and a chromosome's fitness is the inverse of the total cost incurred by the policy that it represents. For our experiments we allow 100 different order-up-to levels, linearly spaced in the range 1–300. Thus each gene has 101 alleles. These parameters were chosen as suitable for the instances we tested.

4 Experiments

We obtained results using several problem parameter settings, and in each case found the same relationships between the algorithms. For this reason, and because of limited space, we present results for only one instance: 100 periods, stationary demands with mean 50 and standard deviation 10 in all periods, and cost parameters $h = 1$, $a = 400$ and $s = 10$. Problems with 100 periods are very hard: none of the methods we test can find the optimal policy within several hours (nor did attempts using Mixed Integer Programming and Reinforcement Learning algorithms). The optimal policy has an expected total cost of 19,561 with replenishment every 4 periods (starting from the first period) and order-up-to levels of 205 deduced from the cyclic nature of the problem (which is not exploited by the algorithms we test).

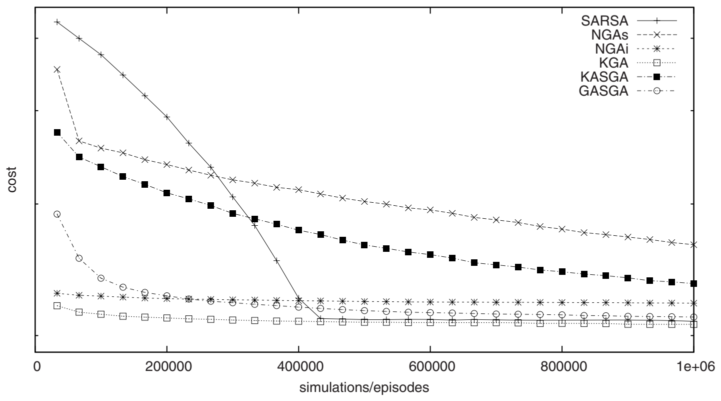
We will compare several GAs using three metrics: the *fitness* of the selected chromosome, the *reliability* of the selected chromosome measured by the number of samples used to compute the fitness, and the *wastefulness* of the GA measured by the number of samples used to estimate the fitness of discarded chromosomes. Almost every chromosome is discarded at some point during search, so the wastefulness is an approximation to the average number of samples used per chromosome. Ideally we aim for a GA with high fitness and reliability, but low wastefulness. In our experiments we aim for a reliability of $U = 1000$. The results are shown in Figure 2.

The fitness graph also shows results for the SARSA(λ) Reinforcement Learning algorithm [24] for comparison, as the problem can be modelled as an episodic Partially Observable Markov Decision Process in which a *state* is the period, an *action* is either the choice of an order-up-to level or the lack of an order (NULL) in a period, and a *reward* (undiscounted) is minus the total cost incurred in a period. We use an ϵ-greedy heuristic, varying ϵ inversely with time as recommended in [24], and tuning the α, λ parameters by the common method of hill-climbing in parameter space. All state-action values were initialised to 0, as the use of optimistic initial values encourages early exploration [24].

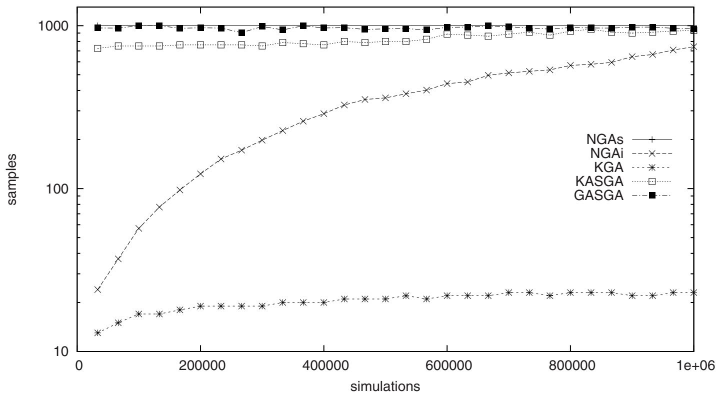
Because there is a range of Pareto-optimal solutions among the chromosomes of a GA, varying from high fitness based on few samples to low fitness based on many samples, we have a problem: how should different GAs be compared? We are interested in fit solutions based on many samples, so for each GA we shall select the chromosome with the greatest value of samples/cost. The results are as follows.

The graphs show that NGAs has high reliability, but it converges quite slowly and has high wastefulness as it uses exactly 1000 samples for every chromosome. NGAi has much better fitness than NGAs. It reaches this fitness rapidly but then make little further progress, perhaps because of its increasing wastefulness. However, it achieves NGAs's reliability by the end of the run, and only matches its wastefulness by the end of the run. Note that the reliability does not quite reach 1000 samples: there is a delay between (i) increasing the number of samples to a given number, and (ii) obtaining a chromosome whose fitness is both high and based on that number of samples. This delay would not occur in a generational GA, in which no chromosome survives to the next generation. We should perhaps use a generational GA to evaluate incremental sampling, as this was the form of GA used in the Noisy GA work, but in this paper our aim is to

fitness:



reliability:



wastefulness:

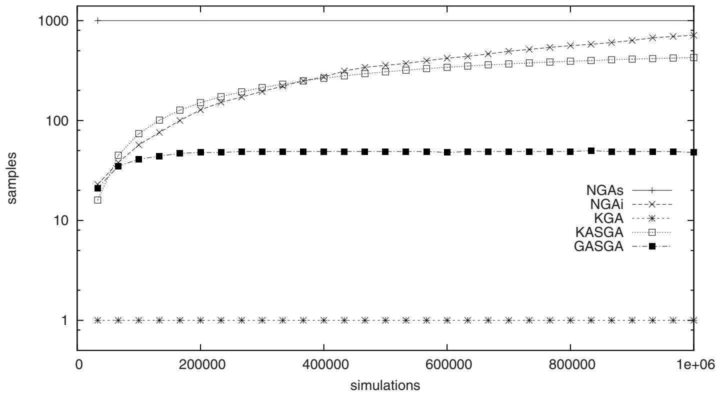


Fig. 2. Experimental results

compare several sampling techniques on the same (steady-state) GA. However, a generational GA will presumably exhibit similar wastefulness.

KGA has excellent fitness but very low reliability. Though KGA has given good results on other problems, here no chromosome survives long enough to achieve a sufficient number of samples. This is caused by the high fitness noise in our problem: as chromosomes are resampled their estimated fitnesses fluctuate significantly, and over many iterations the fittest chromosome is not much more likely to survive than any other. Our problem is very noisy, with the fitness standard deviation not much less than the mean, and KGA seems unsuitable for such problems. KASGA is a marked improvement over KGA. Increasing S until the reliability is approximately 1000 samples, we reach a value $S = 250$. The graphs show that KASGA has better fitness than NGAs but no other algorithm, probably because of its fairly high wastefulness (approximately 400 samples per chromosome). But it does have high reliability, making it more usable than KGA.

GASGA outperforms KASGA and the other algorithms. Again increasing S until reliability is approximately 1000, this time we reach a value of only $S = 25$. The graphs show that GASGA has higher fitness than any other GA (other than KGA). GASGA is also less wasteful than any other GA (other than the unreliable KGA): though it finds high-fitness solutions using 1000 samples, it uses only 39 samples per chromosome on average. This is exactly what we aimed for: a GA that achieves high fitness and reliability but low wastefulness.

As noted above, in further experiments using different problem parameters we obtained the same relationships among the GAs. The only difference was the SARSA(λ) result: on this instance it found a solution that was approximately as good as that found by GASGA, on others it found better solutions, and on others it found worse solutions. This illustrates the known fact that Reinforcement Learning and Evolutionary Computation are rival approaches to some problems, and neither dominates the other over all instances [16].

GASGA should find application to many problems with noisy fitness functions. The required number of samples can be chosen by considering the required solution accuracy and the observed variance in solution fitness. Parameter S must currently be tuned by hand: too small a value causes GASGA to behave like KGA, and it never obtains a reliable solution; too large a value causes it to behave like NGA, and it converges slowly. We tried automating S by maintaining it at a level that only just generates a chromosome with 1000 samples, but this forced it to a higher value than necessary (over 100); automation of S is a topic for future work.

5 Conclusion

We designed a simple new resampling strategy for steady-state GAs that makes no assumptions about fitness noise distributions (though problems with different distributions will probably require the parameter values to be tuned differently), does not require a large population, provides a high level of reliability, yet takes a low number of samples on average. Incorporated into GENITOR and applied to a problem from classical Inventory Control, it gave better results than four other sampling strategies. In future work we will evaluate GASGA on other problems with noisy fitness functions such as perception [7], image registration [9,15], network design [27] and remediation design [11].

None of the algorithms we tested are able to find optimal policies for the inventory problem so it is a challenging benchmark for Evolutionary Computation, and in further experiments we also found it to be hard for Reinforcement Learning and Mixed Integer Programming. This makes it an interesting benchmark despite its simplicity, and in future work we will add features such as order capacity constraints.

References

1. Aizawa, A., Wah, B.: Scheduling of Genetic Algorithms in a Noisy Environment. Evol. Comput. 2(2), 97–122 (1994)
2. Arnold, D.V., Beyer, H.-G.: Local Performance of the (1+1)-ES in a Noisy Environment. IEEE Trans. Evolutionary Computation 6(1), 30–41 (2002)
3. Beielstein, T., Markon, S.: Threshold Selection, Hypothesis Tests, and DOE Methods. In: Congress on Evolutionary Computation, pp. 777–782. IEEE Press, Los Alamitos (2002)
4. Beyer, H.-G.: Evolutionary Algorithms in Noisy Environments: Theoretical Issues and Guidelines for Practice. Computer Methods in Applied Mechanics and Engineering 186(2–4), 239–267 (2000)
5. Branke, J., Schmidt, C., Schmeck, H.: Efficient Fitness Estimation in Noisy Environments. In: Genetic and Evolutionary Computation Conference, pp. 243–250. Morgan Kaufmann, San Francisco (2001)
6. Bui, L.T., Abbass, H.A., Essam, D.: Fitness Inheritance for Noisy Evolutionary Multi-Objective Optimization. In: Genetic and Evolutionary Computation Conference, Washington DC, USA. ACM Press, New York (2005)
7. de Croon, G., van Dartel, M.F., Postma, E.O.: Evolutionary Learning Outperforms Reinforcement Learning on Non-Markovian Tasks. In: Workshop on Memory and Learning Mechanisms in Autonomous Robots, 8th European Conference on Artificial Life, Canterbury, Kent, UK (2005)
8. Darwen, P.J.: Computationally Intensive and Noisy Tasks: Coevolutionary Learning and Temporal Difference Learning on Backgammon. Congress on Evolutionary Computation (2000)
9. Fitzpatrick, J.M., Grefenstette, J.J.: Genetic Algorithms in Noisy Environments. Machine Learning 3, 101–120 (1988)
10. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic Algorithms, Noise, and the Sizing of Populations. Complex Systems 6, 333–362 (1992)
11. Gopalakrishnan, G., Minsker, B.S., Goldberg, D.: Optimal Sampling in a Noisy Genetic Algorithm for Risk-Based Remediation Design. In: World Water and Environmental Resources Congress, ASCE (2001)
12. Hughes, E.J.: Evolutionary Multi-objective Ranking with Uncertainty and Noise. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 329–343. Springer, Heidelberg (2001)
13. Jin, Y., Branke, J.: Evolutionary Optimization in Uncertain Environments — a Survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)

14. Miller, B.L.: Noise, Sampling, and Efficient Genetic Algorithms. PhD thesis, University of Illinois, Urbana-Champaign (1997)
15. Miller, B.L., Goldberg, D.E.: Optimal Sampling for Genetic Algorithms. Intelligent Engineering Systems Through Artificial Neural Networks 6, 291–298 (1996)
16. Moriarty, D.E., Schultz, A.C., Grefenstette, J.J.: Evolutionary Algorithms for Reinforcement Learning. Journal of Artificial Intelligence Research 11, 241–276 (1999)
17. Di Pietro, A., While, L., Barone, L.: Applying Evolutionary Algorithms to Problems With Noisy, Time-Consuming Fitness Functions. In: Congress on Evolutionary Computation, pp. 1254–1261. IEEE, Los Alamitos (2004)
18. Sano, Y., Kita, H.: Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of Search With Test of Estimation. In: Congress on Evolutionary Computation, pp. 360–365. IEEE, Los Alamitos (2002)
19. Schmidt, C., Branke, J., Chick, S.E.: Integrating Techniques from Statistical Ranking into Evolutionary Algorithms. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 752–763. Springer, Heidelberg (2006)
20. Silver, E.A., Pyke, D.F., Peterson, R.: Inventory Management and Production Planning and Scheduling. John-Wiley and Sons, New York (1998)
21. Smalley, J.B., Minsker, B., Goldberg, D.E.: Risk-Based In Situ Bioremediation Design Using a Noisy Genetic Algorithm. Water Resour. Res. 36(10), 3043–3052 (2000)
22. Stagge, P.: Averaging Efficiently in the Presence of Noise. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 188–197. Springer, Heidelberg (1998)
23. Stroud, P.D.: Kalman-Extended Genetic Algorithm for Search in Nonstationary Environments with Noisy Fitness Functions. IEEE Transactions on Evolutionary Computation 5(1), 66–77 (2001)
24. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
25. Then, T.W., Chong, E.K.P.: Genetic Algorithms in Noisy Environments. In: 9th IEEE International Symposium on Intelligent Control, Columbus, Ohio, USA, pp. 225–230 (1994)
26. Whitley, D., Kauth, J.: GENITOR: A Different Genetic Algorithm. In: Rocky Mountain Conference on Artificial Intelligence, Denver, CO, USA, pp. 118–130 (1988)
27. Wu, J., Zheng, C., Chien, C.C., Zheng, L.: A Comparative Study of Monte Carlo Simple Genetic Algorithm and Noisy Genetic Algorithm for Cost-Effective Sampling Network Design Under Uncertainty. Advances in Water Resources 29, 899–911 (2006)

EA-Powered Basin Number Estimation by Means of Preservation and Exploration

Catalin Stoean[1], Mike Preuss[2], Ruxandra Stoean[1], and D. Dumitrescu[3]

[1] Department of Computer Science, Faculty of Mathematics and Computer Science, University of Craiova, Romania
{catalin.stoean,ruxandra.stoean}@inf.ucv.ro
[2] Chair of Algorithm Engineering, Department of Computer Science, Dortmund University of Technology, Germany
mike.preuss@uni-dortmund.de
[3] Department of Computer Science, Faculty of Mathematics and Computer Science, Babes-Bolyai University of Cluj-Napoca, Romania
ddumitr@cs.ubbcluj.ro

Abstract. When using an evolutionary algorithm on an unknown problem, properties like the number of global/local optima must be guessed for properly picking an algorithm and its parameters. It is the aim of current paper to put forward an EA-based method for real-valued optimization to provide an estimate on the number of optima a function exhibits, or at least of the ones that are *in reach* for a certain algorithm configuration, at low cost. We compare against direct clustering methods applied to different stages of evolved populations; interestingly, there is a turning point (in evaluations) after which our method is clearly better, although for very low budgets, the clustering methods have advantages. Consequently, it is argued in favor of further hybridizations.

Keywords: Multimodal optimization, basins of attraction, function optimization, detect-multimodal mechanism.

1 Introduction

The fitness landscape of an optimization problem that is considered for solving by means of evolutionary techniques is almost always completely unknown for the user. Exceptions are represented by the optimization of two- and three-dimensional functions that can be plotted in order to have an idea of the difficulty of the problem at hand. However, for the real-world tasks, one hopes for a unimodal problem, but usually expects that the landscape contains some local optima and one or more global ones. In this respect, it would be very useful to know in advance how multimodal the fitness landscape of the problem is, as this could help decide which optimization algorithm to choose or even set appropriate values for its specific parameters.

The aim of this paper is to design such a tool, also based on an evolutionary algorithm (EA), for the acquisition of data on the profile of the fitness landscape for problems defined over real-valued domains. Instead of obtaining a set of best solutions as usually pursued by contemporary niching EAs, we strive for obtaining an estimate on the number of optima an objective function possesses. One could imagine doing so by simply applying clustering techniques, but even these can only detect different clusters representing optima after somehow progressing towards good regions (as e.g. demonstrated in [1]); a random sample is hard to cluster meaningfully. In order to move into promising areas, some optimization method has to be applied before. However, marching too far e.g. by means of an EA implies the danger of missing several optima on which the subpopulations go extinct. Additionally, randomly initialized recombinative EAs have a natural tendency to contract the population near the search space center, as it has the lowest average distance to all individuals.

We must therefore find a good compromise between basin maintenance, convergence into basins, and further exploration. We track this goal by addressing the topology of the fitness landscape and the preservation of the fittest individuals, in a novel technique tailored after [2]. With this approach at hand, we compare against the straightforward clustering means – chosen as either the state-of-the-art Jarvis-Patrick or the more recent, effective Nearest-Better grouping – with a prior canonical EA for the generation of samples and a final unification of clusters based on the space topology. It goes without saying that the parametrization of any EA based method plays a decisive role for the ability to discover distinct optima and must be taken into account when fitting it for delivering estimates on the multimodality of unknown problems. Different parametrizations will influence the *reachable* search space region of the EA. Consequently, there are no means to perform estimations over areas never visited.

The paper is organized as follows. §2 emphasizes the circumstances and the arguments for the development of such an instrument, while §3 describes the Topological Multimodality Estimator (TME). Conducted experiments to validate and investigate the estimations of proposed technique are outlined in §4; the two clustering algorithms also examine test landscapes and results of expected/found optima of all three are put side by side. Finally, conclusions of the experimentation and outcome are reached.

2 Context and Motivation

When a less-known multimodal problem is considered, one may either resort to *iterated local search* techniques (ILS) [3] or rely on a usually radius-dependent *niching* EA for separating the resembling individuals into different subpopulations (species); in the best case, each one of these would track a different optimum and the number of subpopulations equals the number of optimal solutions. When such an EA is employed, the main concern lies in determining an accurate value for the radius parameter that would help separating the individuals into subpopulations in the most advantageous manner.

Among the radius-related EAs, the most commonly referred is the niching technique of Goldberg and Richardson [4]; for the last two decades, it has represented a source of inspiration for the development of many radius-based EAs in the vein of it [5], [6]. Within this niching technique, individuals are grouped into species by a given radius, so that no distance inbetween them is larger. As previously stated, the value to be chosen for the radius parameter directly depends on the fitness landscape, i.e. on the problem considered for solving. Selecting an appropriate value for the radius assures accurate results. Deb and Goldberg [7] proposed one solution for computing the value for the radius threshold (σ_{share}) that leads to the formation of subpopulations; this has afterwards been embraced by most of the researchers dealing with such parameters. Knowing the number of optima that are to be found, N, and being aware that each niche is enclosed by an n-dimensional hypersphere of radius r, the niche radius σ_{share} can be estimated as $\sigma_{share} = \frac{r}{\sqrt[n]{N}}$.

However, in most of the cases, especially for real-world applications, one usually cannot know in advance the number of optima. Additionally, there is no guarantee that basins of attraction are formed like regular hyperspheres. Methods for approximating the number of solutions for combinatorial optimization problems are described e.g. in [8]. For problems with continuous domain, by investing a small amount of fitness evaluations and using the tool that we put forward, one could have an approximation for N, or at least for the fraction of N that is found relatively often. We assume that this *reachable* fraction of optima heavily depends on the configuration of the underlying EA, so that it is larger for more explorative settings. For any clever technique, a higher number of fitness evaluations invested shall lead to a more accurate estimation, as is the case for the proposed method. Moreover, the present approach also provides approximations of the detected optima, especially for relatively high budgets of evaluations.

3 Topological Multimodality Estimator

As the suggested method represents a pre-processing tool, it shall provide information at a very low cost, i.e. with a reduced budget of fitness evaluations. In order to achieve this aim and, at the same time, explore the search space thoroughly, we utilize a variable sized population: We start with a large population and subsequently continue solely with the most prolific individuals that belong to different basins of attraction. Thus, the number of consumed fitness evaluations is kept low. The population size is allowed to raise again during reproduction, but, unless new basins of attraction are discovered, it is again reduced to a minimum. Another important constraint that had been taken into account is that the technique did not have to require additional parameters (as compared to a canonical EA) that directly depend on the considered problem. In order to avoid the use of a threshold (radius) for subpopulation differentiation, we exploit the topology of the fitness landscape, separating us from simple clustering approaches.

The algorithm begins with the generation of random individuals within the problem domain. A method for detecting whether two individuals belong to the same basin of attraction or not is succeedingly used for selecting the fittest individuals within each of the detected attractors. The procedure was introduced in [9] and called *hill-valley*; taking into account two individuals, it verifies whether there exists either a *hill* (in this case, they track the same peak) or a *valley* (different peaks) inbetween, within the fitness landscape. The method is herein renamed *detect-multimodal* for reasons of simplicity and is described below. From this point on, the search continues only with the fittest individuals that undergo recombination and mutation; obtained offspring are checked to see whether they belong to different basins of attraction than the ones already discovered and the population is updated by retaining the best individual within each attractor. Evolution continues for several cycles until the predefined budget is consumed.

3.1 The Detect-Multimodal Mechanism

The routine takes two individuals (points) as input, checks their relative position within the search space and returns a boolean value, which specifies whether there is a valley between them in the fitness landscape or not: In the latter circumstances, the conclusion is that they climb different hills. In order to reach that decision, a set of interior points between the twois generated. If the fitness of all these is higher than the minimal fitness of the two tested individuals, it is concluded that they track the same optimum. Contrarily, if there exist such a point whose fitness is smaller than the minimal fitness of the two, then it is assessed that they follow different peaks. To conclude, the *detect-multimodal* method verifies the assumption that two individuals track the different optima and returns true if so and false if they follow the same peak [9]. The only required parameter refers to the number of gradations (interior points) taken into account. In all undertaken experiments of current paper, the gradations are values taken equidistantly from the interval [0,1]. The higher the number of interior points, the more precision the outcome of the *detect-multimodal* mechanism has.

3.2 TME Mechanics

TME starts with the initialization of a uniformly randomly generated set of individuals. From this collection, the fittest individual from each different basin of attraction is selected. The chosen individuals undergo an iterative process that includes the following steps. Recombination is applied to the selected pool of individuals: All offspring obtained after recombination are added to the current population which is subject to mutation. The membership of all offspring to the currently detected basins of attraction is verified. For each discovered basin, only the fittest individual is kept. The individuals located in previously unseen basins are also preserved, once more only the fittest one per basin. The selection of the fittest individual within every attraction basin uses the *detect-multimodal* procedure for distinguishing the different attractors (Algorithm 1). The entire population is sorted decreasingly according to fitness. The fittest individual in the population represents a seed. Each individual in the sorted series is considered in turn and checked against the currently found seeds to see if they track distinct optima. If it follows a different peak than all the others that have been tracked until the present moment, then the individual represents a new seed. The recorded seeds (*Seeds*) are taken one by one in terms of Euclidean proximity from the considered individual. It is more

Algorithm 1 Seeds identification

```
Sort population $P$ decreasingly according to fitness;
$Seeds$ = $\{P_1\}$; //the fittest individual automatically becomes a seed
for $i$ = 2 to $n$ do
  Find closest (highest probable) $s \in Seeds$ such that detect-multimodal($P_i$, $s$) = $false$;
  if no such a seed $s$ then
    $Seeds = Seeds \cup \{P_i\}$; // $P_i$ is a seed
  end if
end for
return the $Seeds$ set
```

Algorithm 2. Integration of the newly created individuals

```
for each offspring $x$ in $X$ do
    Find its closest individual $s$ in $Seeds$ for which $detect - multimodal(x, s) = false$;
    if $s$ exists then
        $Seeds = (Seeds\backslash\{s\}) \cup \{fitter(x, s)\}$; // $x$ and $s$ fight for survival
        $X = X \setminus \{x\}$;
    end if
end for
Find the fittest free individual $x$ in $X$;
$NewSeeds = \{x\}$; // $x$ is a new seed
while there are still individuals in $X$ do
    For fittest $x \in$ X find closest $s \in NewSeeds$ such that $detect-multimodal(x, s) = false$;
    if no such seed $s$ then
        $NewSeeds = NewSeeds \cup \{x\}$;
    end if
    $X = X \setminus \{x\}$;
end while
$Seeds = Seeds \cup NewSeeds$; // individuals that follow other peaks are added to population
return the $Seeds$ set with the integrated individuals
```

likely that the individual tracks the same peak as the nearest seed and, consequently, it is verified, by distance rank, against the closest ones to avoid unnecessary calls of the *detect-multimodal* procedure.

The selected set of seeds then enters the evolutionary cycle. Thus, the size of the population is drastically diminished in order to reduce the fitness evaluation cost. Recombination takes place and all resulted offspring are appended to the current population. Hence, the space between the currently tracked optima is explored. Now the whole population undergoes mutation. All obtained offspring from either of the two variation operators are checked against the parent population with two purposes. If an offspring tracks an optimum that has already been followed by one closest individual from the parent population, i.e. the two belong to the same basin of attraction, then only the fitter of the two is kept in the seeds population for the next generation. Secondly, when an offspring lies within a basin of attraction that has not been previously tracked by any other individual from the parent population, it shall be added to the seeds population of the next generation, given that there are no solutions between itself and other descendants that lie within the same basin of attraction (Algorithm 2). *NewSeeds* represents

Algorithm 3. Topological Multimodality Estimator

```
Initialize population and identify the seeds $Seeds$ (Algorithm 1);
while stop condition is not met do
    Apply mating selection to $Seeds$;
    Apply recombination to $Seeds$ and obtain the set of offspring $X$;
    Apply mutation to $Seeds$ and $X$ and append all obtained offspring to $X$;
    Integrate the newly created individuals X to $Seeds$ (Algorithm 2);
end while
return the cardinal of $Seeds$ and the actual solutions in $Seeds$
```

the set of seeds that are detected in the current generation. Each time a new seed is considered for adding, it is checked against the other solutions in the *NewSeeds* set. Finally, *NewSeeds* is appended to the *Seeds* set to form the population that will enter the next generation. The steps of the entire approach are outlined in Algorithm 3. Note that with TME, basin identification in the worst case requires $O(|P|^2)$ extra evaluations ($|P|$ stands for population size). However, this happens only if every newly produced individual is outside all yet identified basins. Additionally, the population is kept small by deleting all non-seed individuals inside a basin. Practical experience shows that with these measures, the process requires rather $O(|P|)$ evaluations for basin testing.

4 Experimental Investigation

Experimentation aims to validate the proposed technique against functions whose number and location of optima is known and compare the performance to that of the cheap alternative of direct clustering on the search space. The Waves function ($F1$, 10 optima) is asymmetric and has some peaks difficult to find as they lie on the border or on flat hills. The Six-Hump Camel Back function ($F2$, 6 optima) exhibits two local optima that are not really higher than their neighboring regions and thus can easily be missed. Additionally, we employ a highly multimodal function (Rastrigin, F3) as model for problems for which neither location nor number of optima is known (it is clear that these are easy to compute in this case). Here, the global optimum is surrounded by a large number of close local optima with small relative differences in their values.

$$\begin{aligned}
\text{F1}(x,y) &= (0.3x)^3 - (y^2 - 4.5y^2)xy - 4.7cos(3x - y^2(2+x))sin(2.5\mathit{\Pi}x)), \\
&\quad -0.9 \le x \le 1.2, -1.2 \le y \le 1.2 \\
\text{F2}(x,y) &= -((4 - 2.1x^2 + \tfrac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2), \\
&\quad -1.9 \le x \le 1.9, -1.1 \le y \le 1.1 \\
\text{F3}(x) &= -(10 \cdot n + \textstyle\sum_{i=1}^{n}(x_i^2 - 10 * cos(2 \cdot \mathit{\Pi} \cdot x_i))), -5.12 \le x_i \le 5.12.
\end{aligned}$$

Two conceptually different nearest neighbor clustering approaches were taken into account for comparison: The Jarvis-Patrick clustering method and the recently proposed Nearest-Better algorithm [10]. A canonical EA evolves a population of individuals for a number of fitness evaluations and clustering is subsequently applied to the final generation. The estimated number of basins is given by the resulting number of clusters, while the approximate optima are given by the prototypes. The Jarvis-Patrick (JP) algorithm [11] considers a list of J nearest neighbors—in terms of (Euclidean) distance—for each individual. Every point in the search space is verified in turn against all others: If the two are contained in each other's neighbor list and have at least K neighbors in common, they are placed in the same cluster. A point cannot belong to more than one cluster. Moreover, if x and y meet the condition to belong to the same cluster and x and z also pass the two criteria, all three will be clustered together, indifferent of the fact of whether y and z also respect the conditions. Finally, the prototypes are determined as the fittest individuals in each cluster. The drawback of this very efficient algorithm consists in the two parameters J and K that results are very dependent upon. The recently introduced Nearest-Better (NB) clustering mechanism relies on the connection to one immediate neighbor for each point, which is also better in terms of fitness—thus, topological information is included in addition to location of points. It essentially assumes that the best individuals in different attraction basins are much further away from each other than the average distance of all individuals to their nearest better neighbors. Every individual connects to its nearest better neighbor (in terms of Euclidean distance once more). The longest edges—those higher than $\phi \cdot mean($lengths of all edges$)$—are removed and the prototypes for each cluster are represented by those individuals that do not connect to others. This approach possesses only one additional parameter to be tuned, with 2 being a good default value according to [10].

Research Questions. How do TME and JP/NB compare in terms of performance on functions with known number of solutions? Can we find a correlation of results obtained for the same configuration but in different runs to estimate the 'reachable' optima?

Pre-experimental Planning. The two selected clustering techniques are utilized on the test cases in advance in order to get acquainted to their behavior. The preceding canonical EA also stops after a fixed number of fitness evaluations. Comparing the number of detected optima against the number of clusters lead to the insight that the two methods largely overestimate the number of attraction basins for both functions, with an advantage on the NB side which is less deceiving. The number of clusters was approx. 3 times higher than the amount of optima. The overrating clustering action of the two techniques had to be resolved, in order to set an equal rival to suggested TME. Ergo, we applied the *detect-multimodal* mechanism with a limited number of interior points (set to 2 in the undertaken experiments) after clusters are determined, in order to unify groups within the same basin. The fitness evaluations employed in this final step are also counted within the totally allowed value.

Task. Directly compare the number of attraction basins found (F1 and F2) or reachable basins estimated (F3) by TME and JP/NB clustering. Measure the ability of the techniques to find the same solutions in multiple runs of the same parameter design.

Experimental Setup. The same budget of fitness evaluations was used both for TME and JP/NB, ranging from 200 to the maximum 2000. The values for all parameters were generated using a Latin Hypersquare Design, i.e. 30 space-filling configurations were produced. The parameters of the evolutionary algorithm were generated within the following intervals for all three methods: Population size is between 2 and 200, mutation and recombination probabilities between 0 and 1 and mutation strength between 0 and 5. Additionally for the JP method, the values for the two parameters J and K were both created between 1 and 25 with the constraint that $J > K$. Plus, as the number of neighbors cannot be higher than the population size, the latter is between 25 and 200. The TME technique also makes use of one parameter, which is the number of interior points considered for the *detect-multimodal* method. The positive integer is generated between 1 and 15. In order to evaluate whether a technique tends to find the same optima in different runs with the same parameter configuration (estimation of *reachable* optima number), we conducted the following computations: Within each parameter design, for every two runs (out of 30 performed here), we use the pair of solution sets A and B for computing the fraction of commonly found (correlated) peaks $d = \frac{|A \cap B|}{|A|}$, where |A| represents the cardinal of the set A. Averaging $d/\mid B \mid$ leads us to an estimator for $1/n$

and thus for n according to [12]. However, we assume that here n is not the number of all optima but rather the number of *reachable* basins for each different configuration. Note that although used with results from 30 runs here, the estimator shall get stable already for very small run sets. Otherwise, it would suffice to simply count the number of optima found within a large number of runs.

Results/Visualization. Table 1 gives the number of optima detected by the three compared techniques on F1 and F2, for the (30 LHD) different algorithm parameter settings. $Best$ columns refer to the highest average number of optima out of all configurations, whether $average$ stands for the average of all runs of all configurations. The small differences between best and average TME results prove the fact that it is not very sensitive to the parameter values.

Table 1. Attraction basins found by TME, JP and NB in the best configuration and average over 30 configurations for $F1$ and $F2$ with different fitness evaluation budgets

Fitness	**F1**						**F2**					
evaluation	**TME**		**JP**		**NB**		**TME**		**JP**		**NB**	
calls	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average
200	5.96	4.48	8.13	5.15	8.36	5.95	3.4	2.97	4	3.33	4	3.37
500	6.96	5.68	8.2	4.33	8.26	5.16	4.43	3.63	4	2.62	4	2.57
1000	8.1	6.71	8.1	2.68	8.56	3.24	5.16	4.08	3.96	1.75	3.83	1.7
2000	9.33	7.89	2	1.07	2.5	1.21	5.63	4.45	1.93	1.22	1.96	1.24

As for the second part of the experiment, which regards the correlation of basin sets, when JP and NB were applied for $F3$ with 5 or more variables, it was found that for several configurations, in all the 30 repeats, the methods found only different attraction basins. In such a case, the value for the estimation n is infinity, which is not a meaningful a priori information about the problem landscape. Consequently, the results of JP and NB for $F3$ with 5 or 10 variables are not reported. For $F1$ with 200 evaluation calls, NB provides the highest averaged value over all configurations for n, 6.08, while for TME this is 5.34 and for JP 5.14. The maximum value in one configuration in these low budget conditions is obtained by NB (8.32). As the number of evaluation calls is increased, the average value for n puts TME in advantage and lowers the values for the clustering methods: For 2000 evaluation calls, the average n for TME is 7.98, while for JP and NB they are 1.26 and 1.4, respectively. The situation is very similar for $F2$, where 3.42 is the estimated reachable basin number for NB, 3.28 when JP is employed, and 3.08 for TME when the lowest budget is used; the value moves up to 4.42 for the highest budget considered for TME, while it goes down for JP and NB towards 1.92 and 1.88. For $F3$ (2 var.), TME has a larger set of solutions that are found in multiple runs, even with 200 evaluations calls: TME has 16.62, NB gives 14.62, while JP has 12.23. The value increases again for TME up to 20.01 for the highest budget, while NB has 3.74 and JP 2.17, in the same circumstances. In case of 5 and 10 variables, TME estimates in average 43.42 and 40.21 solutions (highest budget) and drops to 24.2 (5 var.) and to 17.14 (10 var.), respectively (lowest budget). The configurations with the highest values find n equal to 90.5 (5 var.) and 71.2 (10 var.), while the smallest value in one configuration is around 9 for both cases.

Observations. While JP and NB perform very well for a small number of fitness evaluations and tend to decrease quality as more are considered, TME goes in the opposite direction as a higher budget assures significantly better results. JP and NB with the subsequent *detect-multimodal* manage to overcome the initial overrating. The explanation for the performance decrease in JP and NB for increased number of evaluations lies in the fact that the optimization process drives the population towards one or few optima and, therefore, other local optima are neglected as they remain empty. For both functions $F1$ (10 optima) and $F2$ (6 optima), the closest to the correct solution is TME, when the 2000 evaluation calls are considered. It is interesting to see that JP and NB come very close the real number of basins when the lowest budget is considered (8.36 out of 10 for $F1$ and 4 out of 6 for $F2$). However, it shall also be noted that the difference from the best configuration to the average over all configurations is very high in all cases for the clustering techniques, while there exists only a small such difference for TME. It seems that TME is not very dependent on the values of its parameters, while for JP and NB they play an important role as *wrong* configurations lead to poor results.

Discussion. While JP and NB perform better for lower evaluation costs, TME's accuracy is significantly increased when the evolutionary cycle is prolonged. This is of course due to the interaction between the basin preservation and detection and explorative phases in TME. In JP and NB, this is inexistent and hence, the underlying EA produces smaller and smaller basin numbers while the runs progress. Out of the objective comparison intention of this experiment, further tests were undertaken for TME with a budget of up to 6000. The solutions quality was gradually increased until it reached 9.9 attraction basins for $F1$ and all 6 for $F2$ for the best configuration, in average over 30 repeats. The two presented clustering methods, JP and NB, represent good economical alternatives for estimating the number of different attraction basins of a fitness landscape. However, they strongly depend on the underlying optimization algorithm, so that providing more evaluations does not result in improvements unless the underlying algorithm itself is explorative *and* preserving. However, if such means are provided, as in TME, one gets a much less parameter-dependent and thus robust method, which shall be the better choice, especially for real-world problems. Nevertheless, for higher dimensional spaces, results attest an increased level of parameter-dependency; additional investigations are necessary to observe what parameter settings make TME efficient.

5 Conclusions and Future Work

An evolutionary technique, Topological Multimodality Estimator, to determine the profile of the fitness landscape for real-valued optimization problems, with respect to a low budget of evaluation calls, is introduced. A variable sized population to keep only the most promising solutions for further evolution is thus used. Two improved clustering methods, applied to the set of solutions of a canonical EA, are considered for direct comparison on three multimodal functions. For two dimensions and a very low number of evaluation calls, clustering provided better results. However, when the techniques are allowed to evolve for more generations, TME results improve, while the quality of the EA/clustering method combinations is worse – the underlying EA converges to few attractive regions as it has no means to preserve the already found basins.

Rethinking the obtained results, the conclusion seems obvious. *Simple* clustering methods are cheap and successful especially during the early phases of an optimization, and explorative methods like TME need more evaluations to obtain comparable results, but do have a much higher potential; investing more yields more. Hybridization of TME with one of the clustering methods to eliminate valuable calls of the fitness function, especially at the beginning of the evolutionary cycle, may be a middle alternative. At the same time, clustering affects the quality of solutions when the evaluation calls budget is increased; this would be overcome through the replacement of the canonical EA by the TME engine. Out of the two, NB looks like a better alternative for hybridization, not only because of the better results, but also for the absence of additional parameters. A future TME version should also attain information on the sizes of the detected attraction basins and, finally, the influence of its estimated number of optima on the success probability/convergence rate of a radius-based search algorithm will be investigated.

References

1. Parmee, I.C.: The Maintenance of Search Diversity for Effective Design Space Decomposition Using Cluster-Oriented Genetic Algorithms (COGA) and Multi-Agent Strategies (GAANT). In: 2nd International Conference on Adaptive Computing in Engineering Design and Control (ACEDC 1996), pp. 128–138. University of Plymouth (1996)
2. Stoean, C., Preuss, M., Stoean, R., Dumitrescu, D.: Disburdening the Species Conservation Evolutionary Algorithm of Arguing with Radii. In: Lipson, H. (ed.) 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1420–1427. ACM Press, New York (2007)
3. Ramalhino Lourenco, H., Martin, O., Stützle, T.: Iterated Local Search. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics. Kluwer, Dordrecht (2002)
4. Goldberg, D.E., Richardson, J.: Genetic Algorithms with Sharing for Multimodal Function Optimization. In: 2nd International Conference on Genetic Algorithms, pp. 41–49. Lawrence Erlbaum Associates, Hillsdale (1987)
5. Mahfound, S.W.: Niching Methods for Genetic Algorithms. Technical Report, IlliGAL, 95001, University of Illinois (1995)
6. Beasley, D., Bull, D.R., Martin, R.R.: A Sequential Niche Technique for Multimodal Function Optimisation. J. Evol. Comp. 1(2), 101–125 (1993)
7. Deb, K., Goldberg, D.E.: An Investigation of Niche and Species Formation in Genetic Function Optimization. In: 3rd International Conference on Genetic Algorithms, pp. 42–50. Morgan Kaufman, San Mateo (1989)
8. Reeves, C.R., Eremeev, A.V.: Statistical Analysis of Local Search Landscapes. JORS 55(7), 687–693 (2004)
9. Ursem, R.K.: Multinational Evolutionary Algorithms. In: Congress of Evolutionary Computation (CEC 1999), vol. 3, pp. 1633–1640. IEEE Press, Piscataway (1999)
10. Preuss, M., Schoenemann, L., Emmerich, M.: Counteracting Genetic Drift and Disruptive Recombination in $(\mu \overset{+}{,} \alpha)$-EA on Multimodal Fitness Landscapes. In: Beyer, H.G., O'Reilly, U.M. (eds.) 7th Annual Conference on Genetic and Evolutionary Computation (GECCO 2005), pp. 865–872. ACM Press, New York (2005)
11. Jarvis, R.A., Patrick, E.A.: Clustering Using a Similarity Measure Based on Shared Near Neighbours. J. IEEE Trans. on Comp., 1025–1034 (1973)
12. Jelasity, M., Mike Preuss, M.: On Obtaining Global Information in a Peer-to-Peer Fully Distributed Environment. In: Monien, B., Feldmann, R.L. (eds.) Euro-Par 2002. LNCS, vol. 2400, pp. 573–577. Springer, Heidelberg (2002)

Coevolving Cellular Automata with Memory for Chemical Computing: Boolean Logic Gates in the BZ Reaction

Christopher Stone, Rita Toth, Ben de Lacy Costello, Larry Bull, and Andrew Adamatzky

Unconventional Computing Group
University of the West of England
Bristol BS16 1QY, UK
{Christopher3.Stone,Rita.Toth,Ben.Delacycostello,Larry.Bull, Andrew.Adamatzky}@uwe.ac.uk

Abstract. We propose that the behaviour of non-linear media can be controlled automatically through coevolutionary systems. By extension, forms of unconventional computing, i.e., massively parallel non-linear computers, can be realised by such an approach. In this study a light-sensitive sub-excitable Belousov-Zhabotinsky reaction is controlled using various heterogeneous cellular automata. A checkerboard image comprising of varying light intensity cells is projected onto the surface of a catalyst-loaded gel resulting in rich spatio-temporal chemical wave behaviour. The coevolved cellular automata are shown to be able to control chemical activity through dynamic control of the light intensity. The approach is demonstrated through the creation of a number of simple Boolean logic gates.

1 Introduction

There is growing interest in research into the development of 'non-linear computers'. The aim is to harness the as yet only partially understood intricate dynamics of non-linear media to perform complex 'computations' more effectively than with traditional architectures and to further the understanding of how such systems function. Previous theoretical and experimental studies have shown that reaction-diffusion chemical systems are capable of information processing. Experimental prototypes of reaction-diffusion processors have been used to solve a wide range of computational problems, including image processing, path planning, robot navigation, computational geometry and counting (see [Adamatzky et al., 2005] for an overview). In addition to these applications, Boolean logic gates have been constructed in such excitable chemical systems (e.g., [de Lacy Costello & Adamatzky 2005]) and in bistable systems [Rössler, 1974].

In this paper, we produce networks of non-linear media — reaction-diffusion systems — to achieve user-defined computation in a way that allows direct control of the media. We use a spatially-distributed light-sensitive form of the Belousov-Zhabotinsky (BZ) [Zhaikin & Zhabotinsky, 1970] reaction which supports travelling reaction-diffusion waves and patterns. Exploiting the photoinhibitory property of the reaction, the chemical activity (amount of excitation on the gel) can be controlled by the applied light intensity, namely it can be decreased by illuminating the gel with high light intensity and vice versa. In this way a BZ network is created via light and controlled using cooperative coevolutionary computing to design heterogeneous Cellular Automata (CA) [von Neumann, 1966]. We adapt the chemical system described by Wang et al. [1999] and explore its computational potential based on the movement and control of wave fragments. In our experiments a heterogeneous CA controls the light intensity in the cells of a checkerboard image projected onto the surface of the light sensitive catalyst-loaded gel. Initially a certain number of wave fragments are created on the gel and the coevolved CA is shown able to create a number of two-input Boolean logic gates - AND, NAND and XOR - through dynamic control of the light intensity within each cell in a simulated chemical system.

Previously, several results from the evolution of CAs to perform defined tasks have been presented. Mitchell et al. (e.g., [1993][1994]) have investigated the use of a Genetic Algorithm (GA) [Holland, 1975] to learn the rules of uniform one-dimensional, binary CAs. The GA produces the entries in the update table used by each cell, candidate solutions being evaluated with regard to their degree of success for the given task — density and synchronization. Andre et al. [1999] repeated Mitchell et al.'s work, using Genetic Programming [Koza, 1992] to evolve update rules. They report similar results. Sipper (e.g., [1997]) presented a non-uniform, or heterogeneous, approach to evolving CAs. Each cell of a one- or two-dimensional CA is also viewed as a GA population member, mating only with its lattice neighbours and receiving an individual fitness. He shows an increase in performance over Mitchell et al.'s work by exploiting the potential for spatial heterogeneity in the tasks. In this paper we extend our recently presented version of Sipper's approach to control the behaviour of the BZ system described [Stone et al., 2007].

2 Cooperative Coevolution of Heterogeneous CAs

The characteristics of the chosen chemical system are very much akin to those of two-dimensional cellular automata, such as the Game of Life [Gardner, 1970]. That is, fragments of excitation travel across the surface of the gel, often colliding to form other fragments or self-extinguishing, as do the gliders in "Life." Further, the light projections which cause such behaviour can be arranged in a regular grid of cells over the gel surface. We are therefore interested in using cellular automata to control the behaviour of the fragments to implement computation, particularly forms of collision-based computing (e.g., [Adamatzky, 2002]).

As noted, we have previously presented an approach to the use of a heterogeneous CA to control the BZ chemical system [Stone et al., 2007]. The heterogeneous network has a CA topology, i.e., simple finite automata are arranged in a two-dimensional lattice, with aperiodic boundary conditions (an edge cell has five neighbours, a corner cell has three neighbours, all other cells have eight neighbours each). Each automaton updates its state depending upon its own state and the states of its neighbours. States are updated in parallel and in discrete time. In this work, the transition function of every automaton cell is evolved by a simple evolutionary algorithm (EA).

This approach is very similar to that presented by Sipper [1997]. However, his reliance upon each cell having access to its own fitness means it is not applicable in the majority of chemical computing scenarios we envisage. Instead, fitness is based on emergent global phenomena in our approach (as in [Mitchell et al., 1993], for example). Thus, following Kauffman [1993], we use a simple coevolutionary approach wherein each automaton of the two-dimensional CA controller is developed via a simple genetics-based hillclimber. Due to the use of a single global fitness measure, automata do not evolve in isolation and fitness is influenced by the state of all automata cells in the grid. Hence the automata must coevolve cooperatively to solve the global task.

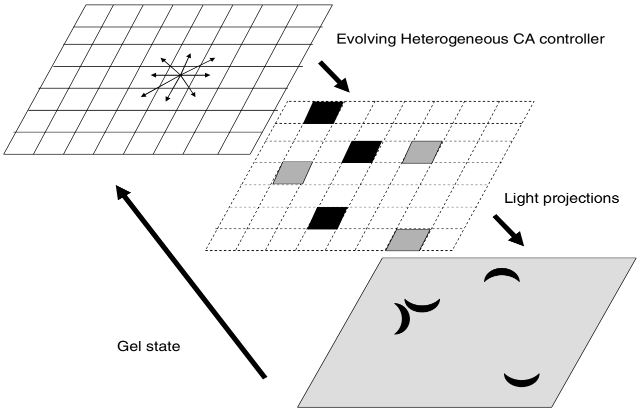


Fig. 1. Relationship between the CA controller, applied grid pattern and chemical system comprising one process control cycle

For a given experiment, a random set of CA rules is created for a two-dimensional array of size 10-by-10, i.e., 100 automata, each responsible for the corresponding area of the gel surface (Figure 1). The transition state for each possible rule for an automaton is represented by a gene in the genome, which takes one of the three discrete light intensity values used in the experiment. As previously mentioned, the grid edges are not connected (i.e., the grid is planar and does not form a toroid) and the neighbourhood size of each cell is of radius 1; cells consider neighbourhoods of varying size depending upon their spatial position. The state of the gel is described by a binary string which indicates the thresholded level of chemical activity in each neighbourhood location. Previously, we showed this system capable of increasing or decreasing the amount of activity across the surface of the gel in both numerical simulation and the actual chemical system [Stone et al., 2007].

3 Chemical Model

Features of the chemical system are simulated using a two-variable Oregonator model modified to account for photochemistry [Field & Noyes, 1973; Krug et al., 1990; Kádár et al., 1997]:

$$\frac{\partial u}{\partial t} = \frac{1}{\square}\left(u - u^2 - (fv + \Phi)\frac{u-q}{u+q} \right) + D_u \nabla^2 u$$

$$\frac{\partial v}{\partial t} = u - v$$

The variables *u* and *v* represent the instantaneous local concentrations of the bromous acid autocatalyst and the oxidized form of the catalyst, $HBrO_2$ and tris (bipyridyl) Ru (III), respectively, scaled to dimensionless quantities. The rate of the photo-induced bromide production is designated by *Φ*, which also denotes the excitability of the system. Low simulated light intensities facilitate excitation while high intensities result in the production of bromide that inhibits the process. The system was integrated using the Euler method with a five-node Laplacian operator, time step Δt=0.001 and grid point spacing Δx=0.62. The diffusion coefficient, D_u, of species *u* was unity, while that of species *v* was set to zero as the catalyst is immobilized in the gel. The kinetic parameters were set to $\varepsilon = 0.11$, $f = 1.1$ and $q = 0.0002$. The medium is oscillatory in the dark which made it possible to initiate waves in a cell by setting its simulated light intensity to zero. At different *Φ* values the medium is excitable, subexcitable or non-excitable. The gel surface area is represented by 200-by-200 simulation points. Parameter settings for the model were experimentally verified against the actual chemical system [Toth et al., 2008].

4 Control Process

Waves were initiated by setting the excitability to zero for a small area under and just outside the bottom centre of the grid. These waves were channelled into the grid and broken up into 12 fragments by choosing an appropriate light pattern as shown in Figure 2(a). The black area represents the excitable medium whilst the white area is non-excitable. After initiation three light levels were used: one is sufficiently high to inhibit the reaction; one is at the sub-excitable threshold such that excitation just manages to propagate; and the other low enough to fully enable it. The modelled chemical system was run for 600 iterations of the simulator. This value was chosen to produce network dynamics similar to those obtained in experiment over 10 seconds of real time.

A colour image was produced by mapping the level of oxidized catalyst at each simulation point into an RGB value. Image processing of the colour image was necessary to determine chemical activity. This was done by differencing successive images on a pixel by pixel basis to create a black and white thresholded image. Each pixel in the black and white image was set to white (corresponding to excitation) if the intensity of the red or blue channels in successive colour images differed by more than 5 out of 256 pixels (1.95%). Pixels at locations not meeting this criterion were set to black. An outline of the grid was superimposed on the black and white images to aid visual analysis of the results.

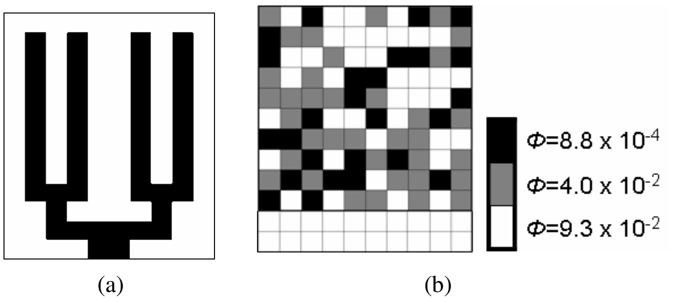


Fig. 2. Showing initiation pattern (a) and a typical example of a coevolved light pattern (b)

The black and white images were then processed to produce a 100-bit description of the grid for the CA. In this description each bit corresponds to a cell and it is set to true if the average level of activity within the given cell is greater than a pre-determined threshold of 10%. Here, activity is computed for each cell as the fraction of white pixels in that cell. This binary description represents a high-level depiction of activity in the BZ network and is used as input to the CA. Once cycle of the CA is performed whereby each cell of the CA considers its own state and that of its neighbours (obtained from the binary state description) to determine the light level to be used for that grid cell in the next time step. Each grid cell may be illuminated with one of three possible light levels. The CA returns a 100-digit trinary action string, each digit of which indicates whether high (Φ=0.093023), sub-excitable threshold (Φ=0.04) or low (Φ=0.000876) intensity light should be projected onto the given cell. The progression of the simulated chemical system, image analysis of its state and operation of the CA to determine the set of new light levels comprises one control cycle of the process. A typical light pattern generated by the CA controller is shown in Figure 2(b).

Another 600 iterations are then simulated with those light-levels projected, etc. until 25 control cycles have passed. After 25 control cycles, the fitness of the emergent behaviour is calculated. As previously mentioned, the EA used in this work employs a single global fitness measure. The nature of the tasks undertaken means that it is not possible to decompose solutions obtained by the EA and apportion fitness to their constituent parts. Instead, a global fitness is determined according to how well the task has been performed and this fitness is assigned to the genome for each CA cell.

The EA is a simple hillclimber. After fitness has been assigned, some proportion of the CA's genes are randomly chosen and mutated. Mutation is the only variation operator used here to modify a given CA cell's transition rule to allow the exploration of alternative light levels for the cell's grid state. For a CA cell with eight neighbours there are 2^9 possible grid state to light level transitions, each of which is a potential mutation site. After the defined number of such mutations has occurred, a generation of the EA is complete and the simulation is reset and repeated as described.

For each cell, the EA keeps track of which grid states are visited since mutation. On the next fitness evaluation (at the end of a further 25 control cycles) mutations in states that were not visited are discarded on the grounds that they have not contributed to the global fitness value and are thus untested. We also performed control experiments with a modified version of the EA to determine the performance of an equivalent random CA controller. This algorithm ignored the fitness of mutants and retained all mutations except those from unvisited states.

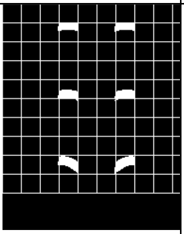
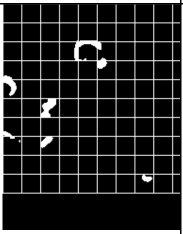
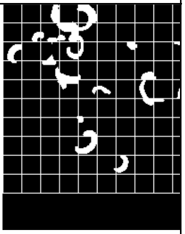
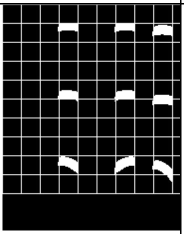
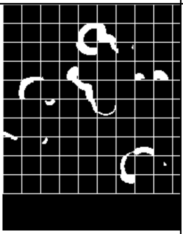
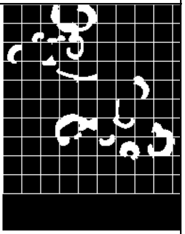
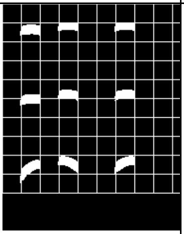
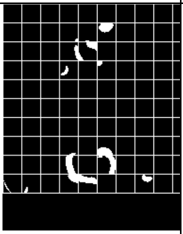
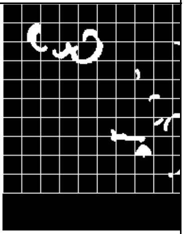
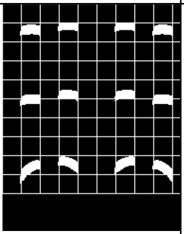
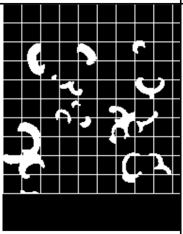
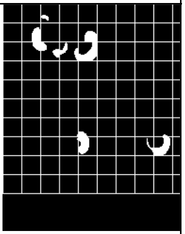
	I_1 I_2		O (AND)		O (NAND)
00		**0**		**1**	
01		**0**		**1**	
10		**0**		**1**	
11		**1**		**0**	

Fig. 3. Typical examples of solutions of AND and NAND logic gates after 25 cycles. Input states I_1, I_2 for the logic gates are shown on the left and consist of two binary digits, spatially encoded using left and right "initiation trees". The EA found the AND solution in 56 input presentations and the NAND in 364 input presentations.

5 Chemical Logic Gates

We have designed a simple scheme to simulate a number of two-input Boolean logic gates under the framework described above where excitation is fed in at the bottom of the grid into the branching pattern. To encode a logical ‘1’ and ‘0’ either both branches or just one branch of the two “trees” shown in Figure 2(a) are allowed to fill with excitation, i.e., the grid is divided into two for the inputs (Figure 3). The number of active cells in the whole grid, that is those with activity at or above the 10% threshold, is used to distinguish between a logical ‘0 and ‘1’ as the output of the system. For example, in the case of XOR, the CA controller must learn to keep the number of active cells below the specified level for the 00 and 11 cases but increase the number for the 01 and 10 case.

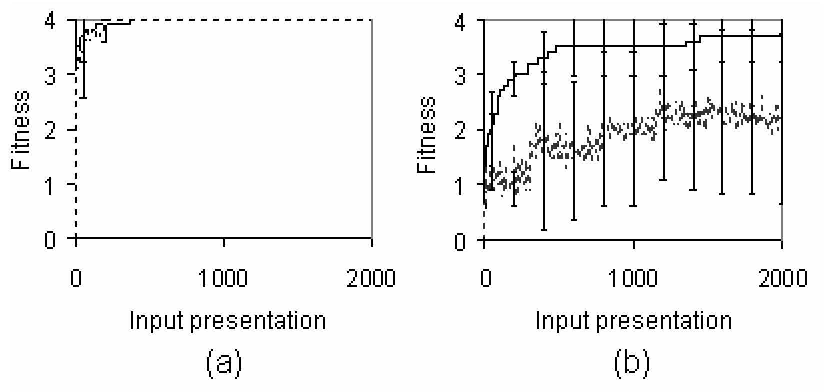


Fig. 4. Showing the average fitness over time for 10 runs for the (a) AND gate and (b) NAND gate on the simulated chemical system. Dashed lines: random controller (10 runs).

Figure 3 shows typical examples of the logic gates learned using the simulated chemical system. Here the mutation rate was set at 4000 genes per EA generation. The required number of active cells was set at 20. Each of the four possible input combinations is presented in turn – 00 to 11 – and for each input presentation the system is allowed to develop for 25 control cycles. Fitness of the logic gate is evaluated after the complete sequence of four input presentations. Each correct output scores 1, resulting in a maximum possible fitness of 4 for a correctly functioning gate. Figure 4(a) and (b) show the fitness averaged over ten runs for the AND and NAND tasks, with similar results found for XOR (not shown).

Table 1 shows a more detailed comparison, namely the results of ten runs for each gate. Due to the high computational requirements needed to perform the simulations a limited number of input presentations were allowed for each experiment and an experiment was considered successful if the controller found a solution within 2000 input presentations. The success rate shows the number of successful runs out of ten. The AND task was so simple that a solution was easily found even with a random controller. This is because the first three inputs provided activity levels similar to the correct outputs, and only the activity levels provided by the 11 input needed to be

Table 1. Performance on AND, NAND and XOR gates

Gate	Controller	Success rate	Min.	Max.	Avg.	Std.
AND	Coevolutionary	10/10	8	144	61	45.69
	Random	10/10	4	200	64	71.73
	Simple Memory	10/10	32	252	140	78.88
	WH Memory	10/10	4	68	34	21.35
NAND	Coevolutionary	7/10	288	>2000	1065	767.21
	Random	4/10	300	>2000	1454	744.49
	Simple Memory	2/10	632	>2000	1858	431.08
	WH Memory	10/10	16	720	228	234
XOR	Coevolutionary	9/10	348	>2000	808	510.08
	Random	10/10	20	1080	455	333.68
	Simple Memory	8/10	48	>2000	1326	784.74
	WH Memory	10/10	8	428	148	131.16

changed to generate appropriate output activity, namely the controller had to increase excitation to get higher than the required number of active cells (that is, those with an activity level greater than or equal to 10%). In contrast, the NAND gate was the most difficult task, because the controller had to achieve the opposite activity levels to those provided by the input states. For 00, 01 and 10 inputs the initial number of fragments were less than the required value so the controller had to increase the excitation to get the correct logical ‘1’ output, while for the 11 input the controller had to decrease the excitation to achieve the logical ‘0’ output. These results indicate the ability of the coevolutionary approach for universal computation since all functions can be constructed by NAND gates. The XOR task was also hard because the activity levels provided by three of the initial inputs (00, 01, and 10) were the opposite of the desired output activity levels and only the 11 input provided an appropriate direct basis for correct output activity.

In the cases where the success rate was less than ten, the averages in Table 1 are the lowest possible averages, since 2000 was taken as the number of input presentations required, even though no solution was found in these cases. For this reason we can only use these data as an indication of the difficulty of the task.

6 Coevolving CAs with Memory

As discussed above, the BZ reaction exhibits rich spatio-temporal behaviour. Recently, the standard CA framework has been extended to explicitly consider temporal dynamics in the transition rule by the inclusion of memory mechanisms (e.g., [Alonso-Sanz, 2004]). Given the strong temporal element of BZ systems, we have explored the utility of including memory within the evolving heterogeneous CA controller.

A simple way of implementing memory is for the CA transition rule Φ to consider the neighbourhood $\mathcal{N}$ of a cell i supplemented with the state σ of the cell on the previous cycle: $\boldsymbol{\sigma_i^{t+1}} = \boldsymbol{\Phi}(\boldsymbol{\sigma_j^t} \in \boldsymbol{N_i}, \boldsymbol{\sigma_i^{t-1}})$. However, this means that the size of the CA's genome must increase to incorporate the extra state and the size of the search space is doubled. To overcome this limitation we have also implemented a form of memory using the well-known Widrow-Hoff Delta rule with learning rate β=0.2. This provides a weighted average memory with no representational overhead in the genome. For this type of memory:

$$m_i^0 = 0.5$$

$$m_i^{t+1} = m_i^t + \beta(\boldsymbol{\sigma_i^t} - m_i^t)$$

$$s_i^t = \begin{cases} 1 & if\ m_i^t > 0.5 \\ 0 & if\ m_i^t \leq 0.5 \end{cases}$$

$$\boldsymbol{\sigma_i^{t+1}} = \boldsymbol{\Phi}(\boldsymbol{s_j^t} \in \boldsymbol{N_i})$$

As Table 1 shows, the simple explicit memory scheme degrades performance but the weighted average scheme improves performance in all cases. Moreover, t-test results with $\alpha = 0.01$ for the NAND and XOR gates show a statistically significant performance improvement when using the Widrow-Hoff memory scheme. Thus, it would appear that the inclusion of memory can enable the CA to capture better the temporal dynamics of the reaction, as envisaged. However, it is apparent from these initial results that factors such as the type and/or depth of memory are important in achieving a benefit.

7 Conclusions

Excitable and oscillating chemical systems have previously been used to solve a number of simple computational tasks. However the experimental design of such systems has typically been non-trivial. In this paper we have presented results from a methodology by which to achieve the complex task of designing such systems — through the use of coevolution. We have shown using a simulated system that it is possible to control the behaviour of a light-sensitive BZ reaction to implement a number of Boolean logic gates. We have also shown that the inclusion of memory within such discrete dynamical systems can better enable them to control such non-linear media. Current work is utilising the actual chemical system (for example, see [Toth et al., 2008]) and exploring the utility of memory mechanisms within CAs to control and model complex systems in general.

Acknowledgements

This work was supported by EPSRC Grants no.'s EP/E049281/1 and GR/T11029/1.

References

Adamatzky, A. (ed.): Collision-based Computing. Springer, Heidelberg (2002)

Adamatzky, A., De Lacy Costello, B., Asai, Y.: Reaction Diffusion Computers. Elsevier, Amsterdam (2005)

Alonso-Sanz, R.: One-dimensional, r=2 cellular automata with memory. International Journal of Bifurcation and Chaos 14, 3217–3248 (2004)

Andre, D., Koza, J.R., Bennett, F.H., Keane, M.: Genetic Programming III. MIT Press, Cambridge (1999)

De Lacy Costello, B., Adamatzky, A.: Experimental implementation of collision-based gates in Belousov–Zhabotinsky medium. Chaos Solitons and Fractals 25, 535–544 (2005)

Field, R.J., Noyes, R.M.: Oscillations in chemical systems. IV. Limit cycle behavior in a model of a real chemical reaction. J. Chem. Phys. 60, 1877–1884 (1973)

Gardner, M.: The fantastic combinations of John Conway's new solitaire game 'Life'. Scientific American 223(4), 120–123 (1970)

Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)

Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution, Oxford (1993)

Koza, J.R.: Genetic Programming. MIT Press, Cambridge (1992)

Krug, H.-J., Pohlmann, L., Kuhnert, L.: Analysis of the Modified Complete Oregonator accounting for oxygen sensitivity and photosensitivity of Belousov-Zhabotinsky systems. J. Phys. Chem. 94, 4862–4866 (1990)

Mitchell, M., Hraber, P., Crutchfield, J.: Revisiting the edge of chaos: Evolving cellular automata to perform computations. Complex Systems 7, 83–130 (1993)

Mitchell, M., Crutchfield, J., Hraber, P.: Evolving cellular automata to perform computations: Mechanisms and impediments. Physica D 75, 361–391 (1994)

Rössler, O.: In: Conrad, M., Güttinger, W., Dal Cin, M. (eds.) Physics and Mathematics of the Nervous System. Springer, Heidelberg (1974)

Sipper, M.: Evolution of Parallel Cellular Machines. Springer, Heidelberg (1997)

Stone, C., Toth, R., Adamatzky, A., Bull, L., De Lacy Costello, B.: Towards the coevolution of cellular automata controllers for chemical computing with the B-Z reaction. In: Thierens, D., et al. (eds.) GECCO 2007: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 472–478. ACM Press, New York (2007)

Toth, R., Stone, C., Adamatzky, A., De Lacy Costello, B., Bull, L.: Dynamic control and information processing in the Belousov-Zhabotinsky reaction using a co-evolutionary algorithm. J. Chem. Phys. (in press, 2008)

Von Neumann, J.: The Theory of Self-Reproducing Automata. University of Illinois (1966)

Wang, J., Kádár, S., Jung, P., Showalter, K.: Noise driven avalanche behavior in subexcitable media. Physical Review Letters 82, 855–858 (1999)

Zaikin, A.N., Zhabotinsky, A.M.: Concentration wave propagation in two-dimensional liquid-phase self-oscillating system. Nature 225, 535–537 (1970)

The Influence of Mutation on Protein-Ligand Docking Optimization: A Locality Analysis

Jorge Tavares, Alexandru-Adrian Tantar, Nouredine Melab, and El-Ghazali Talbi

INRIA Lille - Nord Europe Research Centre
Parc Scientifique de la Haute Borne
59650 Villeneuve d'Ascq, France
jorge.tavares@inria.fr

Abstract. Evolutionary approaches to protein-ligand docking typically use a real-value encoding and mutation operators based on Gaussian and Cauchy distributions. The choice of mutation is important for an efficient algorithm for this problem. We investigate the effect of mutation operators by locality analysis. High locality means that small variations in the genotype imply small variations in the phenotype. Results show that Gaussian-based operators have stronger locality than Cauchy-based ones, especially if an annealing scheme is used to control the variance.

1 Introduction

Protein-ligand docking is an energy minimization search problem with the aim to find the best ligand conformation and orientation relative to the active site of a target protein [1]. The docking problem can be very difficult since the relative orientation and conformations of the two molecules must be considered. Typically, the receptor (usually a protein) is fixed in a three-dimensional coordinate system. By contrast, the ligand can be repositioned and rotated. In case that both receptor and ligand are allowed to be flexible, the problem difficulty increases. As such, the problem is classified, by increasing complexity, into the ensuing categories: rigid-structure docking (both molecules are rigid); rigid protein and flexible ligand; flexible protein and rigid ligand; and, both molecules are flexible. With both molecules flexible, usually the active site of the protein and the ligand, the problem becomes harder. In fact, a higher degree of flexibility implies a considerable increase of the search space size.

For the past years, numerous protein-ligand docking methods have been proposed using different techniques, e.g., incremental construction algorithms, stochastic algorithms and molecular dynamics. For more detailed descriptions, we refer the reader to several review studies [2,3]. Evolutionary and swarm algorithms have recently become one of the dominant search techniques for docking methods and proved to be very successful [3,4]. Although several applications exist, no comprehensive set of studies could be found *to understand why* these algorithms and their components are successful. To the best of our knowledge, the only attempt was made in [5] where several parameters (e.g., population size) and some genetic operators are empirically investigated. When designing an evolutionary approach for this problem, to make it efficient is important to understand its components behavior and effects.

Locality is an important requisite to ensure the efficiency of search and it has been widely studied by the evolutionary computation community [6,7,8]. In general terms, this property indicates that small variations in the genotype space, usually originated by mutation, imply small variations in the phenotype space [6]. A locally strong search algorithm is able to efficiently explore the neighborhood of the current solutions. When this condition is not satisfied, the exploration performed by the algorithm is inefficient and, in a worse case scenario, tends to resemble random search.

The goal of this paper is to perform an empirical locality analysis on the evolutionary algorithm model [9,3] that is usually adopted for protein-ligand docking optimization. Locality measures for the analysis are adopted from the framework proposed by [8] and extended by [10] to deal with real-valued encodings. One distance measure suitable for the selected representation is applied. Mutation is the most frequent operator considered in locality studies. The present study concentrates on the questions: do Gaussian and Cauchy mutation operators have a different effect on phenotypes? Which type of operator is more suitable for evolutionary approaches to protein-ligand? We expect to answer these questions by investigating the impact of the operators on locality. In spite of that, our main research focus is the study of representation properties and the effects of variation operators. The presented work is the first step of a wider study that includes analysis on locality, heritability and heuristic bias.

Results allow us to gain some insights about the degree of locality induced by different mutation operators. The search space is highly multimodal and its shape is influenced by the size, shape and topology of the ligand and the active site being docked [5]. As a consequence of this, even small modifications performed by genetic operators in the structure of an individual lead to large phenotypic changes. An evolutionary algorithm operating on its own is unable to deal with these difficulties. Thus, it is important to know how locality relates to mutation operators commonly used in evolutionary algorithms for molecular docking. Furthermore, understanding the role played by each algorithm's component may provide useful insights for future applications of evolutionary algorithms to this problem.

The rest of the paper is structured as follows. Section 2 contains an overview of the evolutionary algorithm's components used in our experimentation. In section 3 we present the locality analysis and respective discussion. Finally, section 4 contains the main conclusions.

2 Evolutionary Algorithms and Protein-Ligand Docking

Evolutionary algorithms applied to molecular docking can be found since 1993 [11]. A comprehensive review of these efforts, including an outline state-of-the art applications, can be found in [12,3]. One of the most important works is the evolutionary algorithm proposed in [9], commonly referred to as *AutoDock*. This approach is a conformational search method which uses an approximate physical model to evaluate possible protein-ligand conformations. It incorporates flexibility by allowing the ligand to change its conformation during the docking simulation. In addition, pairwise interactions between atoms are pre-calculated, considerably speeding up the docking simulation. To search the space of possible protein-ligand conformations, the approach uses an evolutionary algorithm with a local search method. When this method is applied, the genotype of the individuals is replaced with the new best solution found. This process is usually referred to as Lamarckian evolution.

In our analysis, we adopt an experimental model which uses the main components from [9], because *AutoDock* serves as a basis for the large majority of evolutionary-inspired approaches (e.g.,[3,4]).

2.1 Encoding

During the docking process the protein remains rigid whilst the ligand is flexible. In this case, an individual represents only the ligand. The encoding is an indirect representation. A genotype of a candidate solution is encoded by a vector of real-valued numbers which represent the ligand's translation, orientation and torsion angles [9]. Cartesian coordinates represent the translation, three variables in the vector, whereas four variables defining a quaternion represent the orientation. A quaternion can be considered to be a vector (x, y, z) which specifies an axis of rotation with an angle θ of rotation for this axis. For each flexible torsion angle one variable is used. The phenotype of a candidate solution is composed of the atomic coordinates that represent the three-dimensional structure of the ligand. The atomic structure is built from the translation and orientation coordinates in the ligand crystal structure with the application of the torsion angles.

2.2 Evaluation

To evaluate each individual an energy evaluation function is used. The fitness for each candidate solution is given by the sum of the intermolecular interaction energy between the ligand and the protein, and the intramolecular energy that arises from the ligand itself [9]. An empirical free energy potential composed of five terms is used. The first three terms are pairwise interatomic potentials that account for weal long-range attractive forces and short-range electrostatic repulsive forces. The fourth term measures the unfavorable entropy of a ligand binding due to the restriction of conformational degrees of freedom. The fifth and last term uses a desolvation measure. Further details of the energy terms and how the potential is derived can be found in [9].

2.3 Genetic Operators

Common crossover and mutation operators are applied on the population. In *AutoDock* a standard two-point crossover is used. Cut points only occur between related genes, i.e., separating translational values, orientation values and rotation torsion angles into separate blocks. This is done to avoid disruption of useful parts of the solution [9]. Since the encoding is a real-valued vector, mutation is performed by using evolutionary strategies based operators. The genetic operator acts in the following way: when undergoing mutation, the new value for a gene x' is obtained from the old value x by adding a random real number sampled from a distribution $U(0,1)$:

$$x' = x + \sigma \times U(0,1) \tag{1}$$

The common distribution used for $U(0,1)$ is the standard Gaussian distribution, $N(0,1)$. In spite of that, the *AutoDock* approach replaces the Gaussian distribution with a Cauchy distribution:

$$C(x,\alpha,\beta) = \frac{\beta}{\pi\beta^2 + (x-\alpha)^2} \tag{2}$$

where $\alpha \leq 0, \beta > 0, -\infty < x < +\infty$ (α and β are parameters that control the mean and spread of the distribution). The Cauchy distribution has a bias toward small variations. However, unlike the Gaussian distribution, it has thick tails which allows larger variations more frequently. Some evolutionary approaches to molecular docking use both distributions for mutation operators, e.g., [5].

One important aspect is the value for the parameter σ. If it is set too low, exploitation overcomes exploration and if set too high *vice versa*. The value can be fixed or self-adapted (e.g., if an evolutionary strategy approach is used). In [5], annealing schemes to control σ as a function of time, i.e., the number of generations are proposed. Results show that the following scheme presents good results, scaled with 0.1:

$$\sigma(t) = \frac{1}{\sqrt{1+t}} \tag{3}$$

We also include in the analysis the simple uniform mutation operator. It works in the following way: when applied to a gene, it assigns a new random value according to the gene bounds, sampled from a standard uniform distribution. This operator serves as a comparison baseline.

3 Locality Analysis

We selected several instances from the *AutoDock* test suite to perform the locality analysis. Due to space limitations, we will only present results obtained with the HIV-1 protease/XK 263 protein-ligand complex. It has 10 rotatable bounds with 8 torsional degrees of freedom and is one of the largest complexes in the suite. Results obtained with other instances (e.g., β-Trypsin/benzamidine) follow the same pattern. The parameter σ is set to a value of 0.1 which previous studies in computational chemistry problems have shown to be a good value [10].

3.1 Related Work

Several techniques have been proposed to estimate and study the behavior of evolutionary algorithms and their components. Some of these methods adopt measures that are, to some extent, similar to the locality property. We highlight the most relevant ones.

The concept of fitness landscapes, originally proposed by [13], establishes a connection between candidate solutions and their fitness. Moreover, [14] proposed fitness distance correlation as a way to determine the relation between fitness and distance to the optimum. If fitness values increase as the distance to the optimum decreases, then search is expected to be easy. An alternative way to analyze the fitness landscape is to determine its ruggedness. In [15], it is proposed the adoption of autocorrelation functions to measure the correlation of all points in the search space at a given distance. In [6], conditions for strong causality are studied. A search process is said to be locally strong causal if small variations in the genotype space imply small variations in the phenotype space. In this case, variations in genotypes are caused by mutation.

3.2 Definitions

Investigations with an evolutionary framework usually means considering two spaces: the genotype space Φ_g and the phenotype space Φ_p. Genetic operators are applied on Φ_g while the fitness function, f, is applied to solutions from the phenotype space: $f : \Phi_p \rightarrow \Re$. To establish the similarity between two individuals from Φ_p a phenotypic distance has to be defined. This measure captures the semantic difference between two solutions and is directly related to the problem being solved. The phenotypic distance can be determined with a structural distance measure. To evaluate a final ligand conformation we compare it with the experimental structures using the standard Cartesian root-mean-square deviation (RMSD):

$$RMSD_{lig} = \sqrt{\frac{\sum_{i=1}^{n} dx_i^2 + dy_i^2 + dz_i^2}{n}} \tag{4}$$

where n is the number of atoms in the comparison and dx_i^2, dy_i^2 and dz_i^2 are the deviations between the crystallographic structure and the corresponding coordinates from the predicted structure lig on Cartesian coordinate i. RMSD values below or near 1.5Å can be considered to be a success criterion. Thus, lower values mean that the observed and the predicted structures are similar. Therefore, our structural distance measure determines the difference between RMSD values of two phenotypes:

$$d_{struct}(A, B) = |RMSD_A - RMSD_B| \tag{5}$$

We adopt the innovation measure proposed by Raidl and Gottlieb [8] to study the effect of mutation on locality. To predict the effect of applying this operator we use the distance between individuals in a mutation step. Let X be a solution and X^m the result of applying m mutation steps to X, then the Mutation Innovation (MI) is given by:

$$MI = dist(X, X^m) \tag{6}$$

MI illustrates how much innovation the mutation operator introduces, i.e., it aims to determine how much this operator modifies the semantic properties of an individual. Locality is directly related to this measure. The application of a locally strong operator implies a small modification in the phenotype of an individual. The distance between the two solutions is small. On the other hand, operators with weak locality allow large jumps on the search space. To evaluate MI, 1000 random individuals are generated. Afterwards, a sequence of mutation steps is applied to each one of them and the distance between the original individual and the new solution is measured. In our experimentation, we start by applying a single mutation step. Later, we repeat the experiment with k successive mutation steps, with $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$.

3.3 Experimentation and Discussion

Table 1 shows the characteristic values for MI with a single mutation ($k = 1$). $P(MI = 0)$ represent the percentage of cases for which $MI = 0$. $E(MI|MI > 0)$ and $\sigma(MI|MI > 0)$ show the mean value and the standard deviation of MI, for $MI > 0$. They act as estimations for the expected values. $Max(MI)$ gives the maximum value for MI.

Table 1. Characteristic values for the Mutation Innovation MI with $k = 1$

	Uniform	Gaussian 0.1	Cauchy 0.1	Gaussian AS	Cauchy AS
$P(MI = 0)[\%]$	0.30	7.30	4.70	9.10	7.40
$E(MI\|MI > 0)$	1.28	0.04	0.15	0.03	0.11
$\sigma(MI\|MI > 0)$	1.71	0.11	0.52	0.08	0.39
$Max(MI)$	7.49	1.19	5.94	0.75	5.54

We start by considering the case where mutation does not affect the phenotype, $MI = 0$ (occurring with probability $P(MI = 0)$). A large value of $P(MI = 0)$ indicates that mutation does not make often moves in the search space. In alternative, it may also be an evidence of redundancy or strong heuristic bias since many elements could map to the same phenotype. Table 1 shows that this is not the case. The probability of $MI = 0$ is low for every operator. Uniform mutation displays the lowest value (0.30) compared to Gaussian and Cauchy mutation. Since this operator replaces a complete gene in opposition to performing a small modification, this modification is enough to produce a new phenotype. For Gaussian and Cauchy operators the final result in behavior is similar. The modifications operated by these distributions will produce different phenotypes although the probability of generating a number that is small

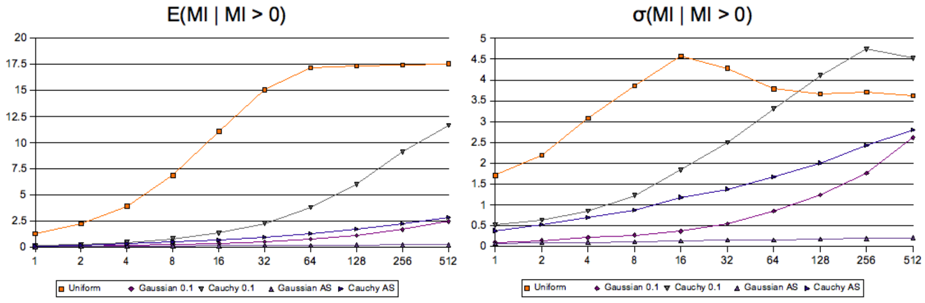


Fig. 1. $E(MI|MI > 0)$ and $\sigma(MI|MI > 0)$ over the number of mutations

enough to induce the same individual is slightly larger. The small difference between Cauchy and Gaussian mutation is explained by the thick tails of the Cauchy distribution. These allow larger variations more frequently than with Gaussian distribution and as such, lower its $P(MI = 0)$.

Moving on to $E(MI|MI > 0)$, $\sigma(MI|MI > 0)$ and $Max(MI)$, in general, small values indicate high locality. A single mutation changes the phenotype only a little and thus, should be aspired [8]. Although lower values are good signs for a *good* locality, it should be noted that larger values for the standard deviation and for Max of MI may not necessarily be a bad indication. In our case, both distributions show low values for the locality measures. However, Cauchy mutation operators present larger values. For example, $E(MI|MI > 0)$ displays 0.15 and 0.11 in comparison to 0.04 and 0.03. The same pattern is observed for the remaining measures. To establish if these differences are statistically significant, we performed the Wilcoxon rank sum test with significance value $\alpha = 0.01$. We found significant differences between Gaussian and Cauchy mutation operators, with fixed and annealing schemes. Differences between operators with the same distribution were not found (e.g., Gaussian with fixed variance and Gaussian with annealing scheme).

A Gaussian operator displays better locality properties but, how does the distribution of mutation innovation changes when considering $k > 1$ mutations? We will now consider the case for $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$. Figure 1 plots the empirically obtained mean values $E(MI|MI > 0)$ and $\sigma(MI|MI > 0)$ over the number of mutations k. Cauchy mutation without the annealing scheme shows higher mean and standard deviations than Gaussian operators and the Cauchy operator with the annealing scheme. For values of k larger than 32, the difference between this operator and the others increases considerably. This indicates weak locality with respect to the Cauchy operators. However, uniform mutation displays much higher values. When looking at the $E(MI|MI > 0)$ values, uniform mutation starts to express much larger values from $k = 1$, only stabilizing around $k = 64$. Nevertheless, the difference is very high showing the low locality properties induced by this operator. The combination of a Gaussian distribution and the annealing scheme displays the best behavior: for all the

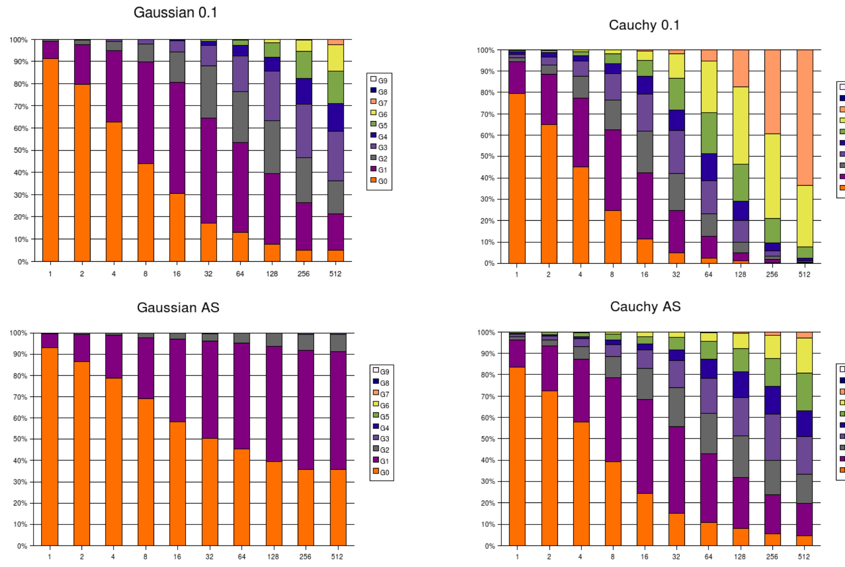


Fig. 2. Distribution of structural distances for $k \geq 1$ mutations

mutation steps the mean and standard values remain low. This suggests a strong locality effect for this operator. The same is also true for the Gaussian operator with fixed variance and the Cauchy operator with annealing scheme operators. Nevertheless, for larger values of k, these two operators start to display a small $E(MI|MI > 0)$ increase.

Regarding $\sigma(MI|MI > 0)$ values, the pattern is similar but some remarks must be made. The most stable operator is Gaussian with annealing scheme whereas the most unstable are uniform mutation and Cauchy with fixed variance. The Cauchy operator starts with low standard deviation values but there is a shift of phase near $k = \{16, 32\}$. From this point on, the standard deviation values rise. For $k = 128$ the values are larger than uniform mutation. This is consistent with the mean values since by this time, uniform mutation has stabilized, although the distance between the mutated individuals and the originals is very large. At this point there is no semantic relation between the individuals. The Cauchy operator follows the same behavior. Here, the loss of semantic relationship occurs later in the process.

Grouping the distances between the original solution and the successive mutants allow us to observe the different types of changes operated by mutation for $E(MI|MI > 0)$. Given a structural distance d_{struct} between two phenotypes, the set G_i to which d_{struct} is assigned is determined the following way: $\{G0 : 0 \leq d_{struct} < 0.1; G1 : 0.1 \leq d_{struct} < 0.5; G2 : 0.5 \leq d_{struct} < 1; G3 : 1 \leq d_{struct} < 2; G4 : 2 \leq d_{struct} < 3; G5 : 3 \leq d_{struct} < 5; G6 : 5 \leq d_{struct} < 10; G7 : 10 \leq d_{struct} < 25; G8 : 25 \leq d_{struct} < 50; G9 : 50 \leq d_{struct}\}$. The specific values that were selected to determine intervals are arbitrary. The relevant information is the distribution of the structural distances through the sets. Low order sets (i.e., small variations) suggest that locality is strong.

The charts in figure 2 show the distribution of structural distances for 1000 individuals, for all operators variants, with each column representing a mutation step. Important differences can be observed. Gaussian operators exhibit high locality: with the annealing scheme, $\geq 50\%$ of the distances belong to the first group until $k = 32$. From $k = 32$ to $k = 512$, the percentage of distances in the first groups stabilizes around 35%. Moreover, the majority of the remaining distances are within the lower three groups. This pattern is not observed elsewhere. This shows that this operator preserves the semantic properties of the individuals subject to mutation well. The Gaussian operator with fixed variance and the Cauchy operators demonstrate similar distributions. The main difference is given by Cauchy with fixed variance. The loss of the semantic properties can clearly be seen from the last four columns (representing the mutation steps for large k). Here the amount of individuals belonging to the last groups is considerable. This supports our previous plot analysis of $\sigma(MI|MI > 0)$.

4 Conclusions

Most evolutionary algorithms applied to this problem use one of these distribution operators (or variants based on them) but mostly Cauchy-based. However, no studies were performed to conclude about its efficiency and performance with the exception of [5]. The Gaussian operator with the annealing scheme is reported to attain the best optimization results. Nevertheless, an investigation on *why* the operator is able to achieve these results is not provided. Since Cauchy-based operators are commonly used in evolutionary approaches to molecular docking, it is important to understand their behavior and related operators.

We investigated the degree of locality induced by different mutation operators when applied to protein-ligand docking optimization. Results confirm that high locality is important and explain the behavior of different mutation operators. As such, the useful outcome from this work is twofold: 1) it explains in terms of locality the operators under investigation; 2) it provides hints on how future mutation operators can be developed. Is important for an operator to induce strong locality to obtain good optimization results. This result is sustained by the study described in [5] and experimentation performed by us (not shown due to space constraints). Gaussian mutation provides locally strong operators and this is especially true when used in conjunction with an annealing scheme. This is an indication that more fine-tuning of the conformations is allowed. On the other hand, Cauchy-based operators show a lesser degree of locality. The operator with the annealing scheme shows a locality similar to Gaussian mutation with fixed variance. Thus, these operators can provide a more exploratory role. In fact, the higher locality shown by Gaussian mutation with the annealing scheme could prove to be excessive, and therefore, difficulties to overcome traps in the search space could arise. Although results from optimization runs show this operator obtaining the best results [5], there are other algorithm's components which also have a direct influence on the search process. As such, it is necessary to extend our research to other components, e.g., crossover, and perform additional optimization runs. Finally, in this work, local search methods were not considered. These techniques will be the focus of a future publication since the impact of local search is an important aspect of an evolutionary algorithm. As future research, we will extend this study to heritability and heuristic bias properties, to study the effects of representation and operators on this problem.

References

1. Neumaier, A.: Molecular modeling of proteins and mathematical prediction of protein structure. SIAM Review 39, 407–460 (1997)
2. Morris, G.M., Olson, A.J., Goodsell, D.S.: Protein-ligand docking. In: Clark, D.E. (ed.) Evolutionary Algorithms in Molecular Design, pp. 31–48. Wiley-VCH (2000)
3. Thomsen, R.: Protein-ligand docking with evolutionary algorithms. In: Fogel, G.B., Corne, D.W., Pan, Y. (eds.) Computational Intelligence in Bioinformatics, pp. 169–195. Wiley-IEEE Press, Chichester (2008)
4. Korb, O., Stützle, T., Exner, T.: An ant colony optimization approach to flexible protein-ligand docking. Swarm Intelligence 1, 115–134 (2007)
5. Thomsen, R.: Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids. Biosystems 72, 57–73 (2003)
6. Sendhoff, B., Kreutz, M., Seelen, W.V.: A condition for the genotype-phenotype mapping: Casualty. In: 7th Int. Conf. on Genetic Algorithms, pp. 73–80 (1997)
7. Rothlauf, F.: On the locality of representations. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003), pp. 1608–1609 (2003)
8. Raidl, G.R., Gottlieb, J.: Empirical analysis of locality heriability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. Evolutionary Computation Journal 13, 441–475 (2005)
9. Morris, G.M., Goodsell, D.S., Halliday, R.S., Huey, R., Hart, W.E., Belew, R.K., Olson, A.J.: Automated docking using a lamarckian genetic algorithm and and empirical binding free energy function. Journal of Computational Chemistry 19, 1639–1662 (1998)
10. Pereira, F.B., Marques, J., Leitão, T., Tavares, J.: Analysis of locality in hybrid evolutionary cluster optimization. In: Proceedings of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, Canada, pp. 8049–8056. IEEE Press, Los Alamitos (2006)
11. Dixon, J.S.: Flexible docking of ligands to receptor sites using genetic algorithms. In: Proc. of the 9th European Symposium on Structure-Activity Relationships, Leiden, The Netherlands, pp. 412–413. ESCOM Science Publishers (1993)
12. Moitessier, N., Englebienne, P., Lee, D., Lawandi, J., Corbeil, C.: Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go. British Journal of Pharmacology 153, 1–20 (2007)
13. Wright, S.: The roles of mutation, inbreeding, crossbreeding and selection in evolution. In: Proceedings of the VI International Conference on Genetics, vol. 1, pp. 356–366 (1932)
14. Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, University of New Mexico, Albuquerque, New Mexico (1995)
15. Weinberger, E.D.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics 63, 325–336 (1990)

Testing the CAX on a Real-World Problem and Other Benchmarks

A. Tchougang[1], A. Blansché[1], L.A. Baumes[2], N. Lachiche[1], and P. Collet[1]

[1] Université Louis Pasteur, LSIIT, FDBT, Pôle API, F-67400 Illkirch, France
[2] Universidad Politecnica de Valencia (UPV-CSIC), Instituto de Tecnologia Quimica, Av. de Los Naranjos, E-46022 Valencia, Spain
ariel.tchougang-njomou@ulp.u-strasbg.fr, baumesl@itq.upv.es, {lachiche,alexandre.blansche,pierre.collet}@lsiit.u-strasbg.fr

Abstract. Intrigued by the interesting Context Aware Crossover (CAX) operator proposed by Majeed and Ryan in a number of recent papers, this operator was tried on a real problem (solid catalysts optimisation) where unfortunately, no improvement was detected. An implementation of the benchmarks used in [MR06b] seems to show that the CAX is mostly an exploitation operator, boosting the average fitness of the population.

1 Introduction

At EuroGP'06 in Budapest, Majeed and Ryan presented a paper [MR06a] (nominated for the best paper award) on a new Genetic Programming crossover operator called *Context Aware Crossover* (CAX) that (according to the paper) yielded great results on several usual benchmarks including symbolic regression, that were confirmed in at least three other papers [MR06b, MR07b, MR07a].

It was therefore very tempting to try it out on the real problem of catalyst optimisation, which is a form of multi-objective symbolic regression. Unfortunately, the results were not as good as expected, so we tried to reproduce the benchmarks of [MR06b] in order to study this operator more thoroughly.

This paper starts with a quick description of the CAX crossover, followed with a presentation of the results obtained on the catalyst optimisation problem. Then, some benchmarks are presented to compare the CAX with the standard GP crossover, based on consumed CPU-time, and a conclusion ends the paper.

2 Quick Description of the Context Aware Crossover

The CAX aims at improving the efficiency of the standard GP crossover (which is generally regarded as mostly destructive, even though it is the main drive of Genetic Programming), by improving the second part of the operation, *i.e.* choosing where to graft into parent 1 a subtree chosen in parent 2.

Usually, a "modern" GP crossover operator creates one new child from two selected parents (P_1 and P_2) by:

1. Randomly selecting a subtree S_2 in P_2 (with 90% chance to select a node),
2. Randomly selecting a subtree S_1 in P_1 (pointing on a node if S_2 is a node),
3. Creating a child which is the clone of P_1 with subtree S_2 in place of S_1.

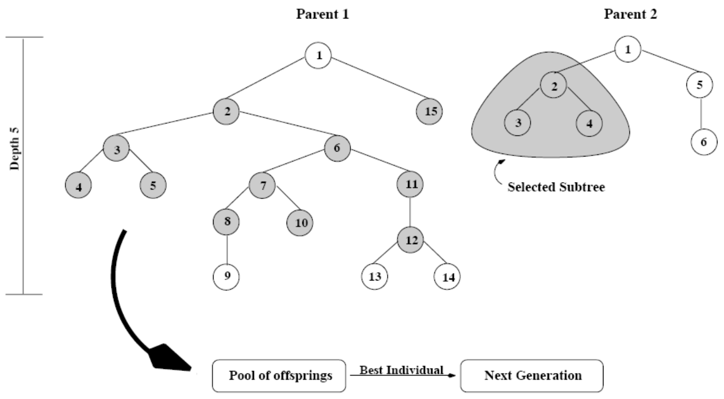


Fig. 1. Context Aware Crossover: the shaded nodes in P_1 are possible crossover points where the selected subtree S_2 from P_2 can go in. This figure is extracted from [MR06b].

In the case of the CAX operator, after selecting S_2 in P_2 like in step 1 above, one tries to find the best place where it could be grafted in $P1$. All nodes of P_1 can potentially receive the graft, excluding:

1. The root of P_1 (that cannot be a crossover point, otherwise, the entire individual would be replaced by the S_2 subtree).
2. Nodes in the bottom of P_1, where grafting of S_2 would result in violating the maximum depth constraint (depth 5 in fig. 1).

All possibilities are deterministically explored, by evaluating all possible children resulting from the graft of S_2 wherever P_1 can receive it (gray nodes in fig. 1). The CAX returns the child with the best fitness.

Even though the exhaustive exploration of all potential crossover points in P_1 is clearly expensive, Majeed and Ryan claimed exceptional results in their different papers, convincing us to try this new operator on the real world problem of heterogeneous catalysts optimisation.

3 Tests on a Real World Problem: Catalyst Optimisation

Catalytic processes constitute the fundamentals of modern chemical and petrochemical industries. Over 70% of the current chemical processes are catalytic, whereas for the newly introduced ones this percentage is over 90. In highly developed industrial countries, catalytic processes create about 20% of the Gross Domestic Product (GDP).

Here, the chosen application [SBMC08] deals with catalysts for obtaining long chain aliphatic epoxides that can be functionalised for application in lubricants, plasticizers, chemicals and fine chemicals production. The chemical reaction is the epoxidation of 4-decene with organic peroxides (*cf.* fig. 2).

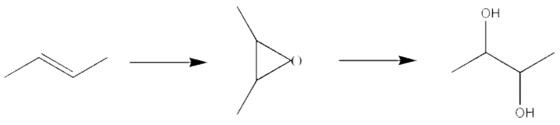


Fig. 2. Epoxidation reaction: the molecule in the centre is the desired product while the one on the right hand side is undesirable

Progressing from catalyst design and discovery up to commercial applications involves several steps and iteration loops. In general, the overall process may take 15 to 20 years. Drastic and successful changes occurred in the 90's *via* fast synthesis and screening of large libraries of diverse formulations by using fully automated working stations and analytics. Combinatorial heterogeneous catalysis [AMS$^+$06, CDCJ$^+$06] is a multi- and trans-disciplinary field, as it requires the intensive support of robotics, physics, mechanical and electrical engineering but also statistics [BMC07], data mining [KFB$^+$04, Bau06], artificial intelligence [BFLM04] and many more.

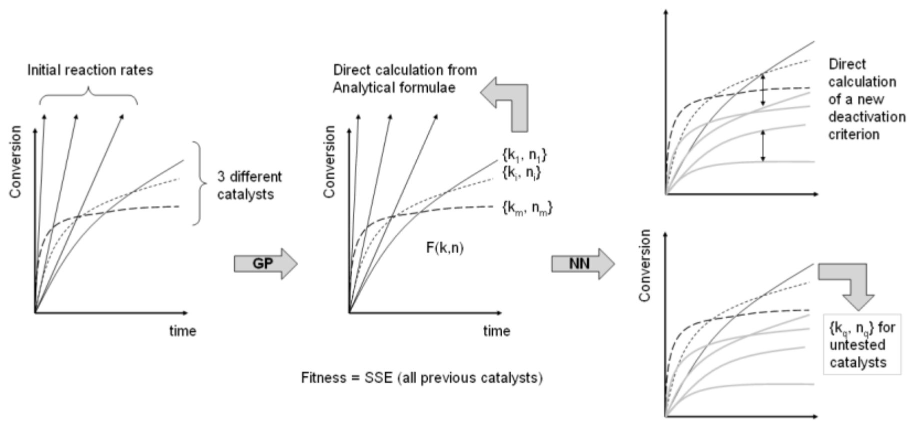

Initial reaction rates
Conversion
3 different catalysts
time
GP
Direct calculation from Analytical formulae
Conversion
{k1, n1}
{ki, ni}
{km, nm}
F(k,n)
time
Fitness = SSE (all previous catalysts)
NN
Direct calculation of a new deactivation criterion
Conversion
{kq, nq} for untested catalysts

Fig. 3. General scheme: the use of Genetic Programming enables the direct calculation of a deactivation criterion and a prevision through neural network, both based on the parameters of the best function found by GP

The aim of catalysts optimisation using Genetic Programming is multiple. Firstly, the initial reaction rates, noted r, are usually calculated manually due to the lack of analytical functions describing the evolution of the formation of products under the action of the catalysts. Therefore, such a laborious task can be easily handled if a function is available, and a map of the entire search space can be found using a given machine learning approach, here a Neural Network.

However, optimising catalyst performance does not only consider the initial reaction rate but also the behaviour the catalyst under a given period of reaction, so-called deactivation, which requires the whole series. In order to find the optimal set of catalysts based on all previous experiments and taking into account both criteria, the following strategy has been employed: the synthesis variables of the catalysts have been correlated with the parameters of the best function found by the GP (*cf.* fig. 3).

Due to the limited amount of experimental data and in order to discard overfitting of the NN, the number of parameters of the function must be minimized. Thus, a multi-objective GP has been implemented whose aims are to minimize both the squared error along the entire series for each catalyst, and the number of parameters of the function.

3.1 Description of the Input Data

The dataset is composed of 148 different synthesized and tested catalysts. Catalysts activity is monitored during 16 hours, and for each one, a series of seven conversion measurements is obtained, *i.e.* the quantity of initial reactant which is transformed along time. Since catalytic activity decreases over time, all curves share a general shape, characterized by a positive first derivative and a negative second derivative. The aim of the GP algorithm is to discover the function behind the general shape (corresponding to all catalysts).

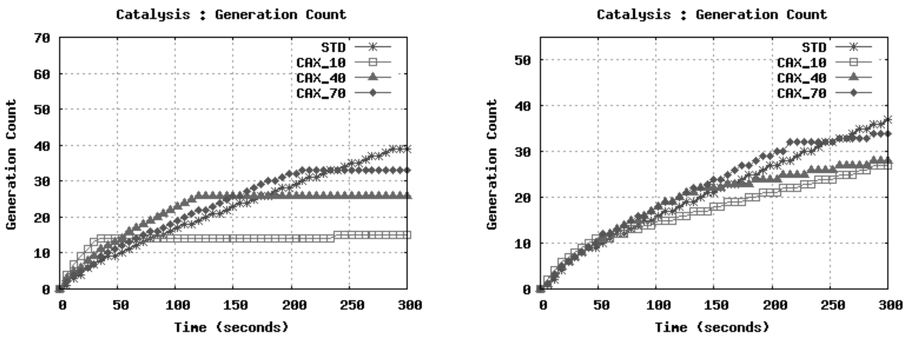


Fig. 4. Catalyst optimisation problem. Number of generations for constant population (left) and reduced population for CAX (right). Each run takes around 13 hours on a 3Ghz PC (5 minutes evolution $\times$ 148).

3.2 Results with and without CAX on Catalyst Optimisation

This difficult problem was first tackled with a tailored GP algorithm that did not use the CAX operator. The adjusted fitness (in the Koza sense[1]) of the best individual measured on the evaluation set is 0.93 (for each catalyst, data is divided in a learning set, a test set to detect overfitting and an evaluation set).

[1] $\texttt{adjusted\_fitness} = 1/(1 + \texttt{raw\_fitness})$

Then, the CAX was tried, but with bad results, which was quite disappointing. Coding was carefully checked to no avail. In their different papers, Majeed and Ryan suggest to first use the standard GP crossover, and then start the CAX only after some time, so curves were plotted for CAX_10 (CAX started after 10% of the run), CAX_40, CAX_70 and no CAX (*cf.* fig. 4).

In [MR06b], the authors say that they were giving an "advantage" to standard GP by giving it 4,000 individuals where the algorithm using CAX only needed 200. In fact, tests showed that if the same population size is used for standard GP and CAX, the generation count just freezes when the CAX starts, due to the huge amount of children evaluations that this operator needs (*cf.* fig. 4 left). It therefore appears that using a population of 200 individuals only for CAX was in fact giving an advantage to CAX rather than GP...

If the bad results were due to the lack of evolution, it was decided to give the CAX an advantage by reducing its population by 95% when it starts, so as to keep a generation count roughly equivalent to standard GP (*cf.* fig. 4 right).

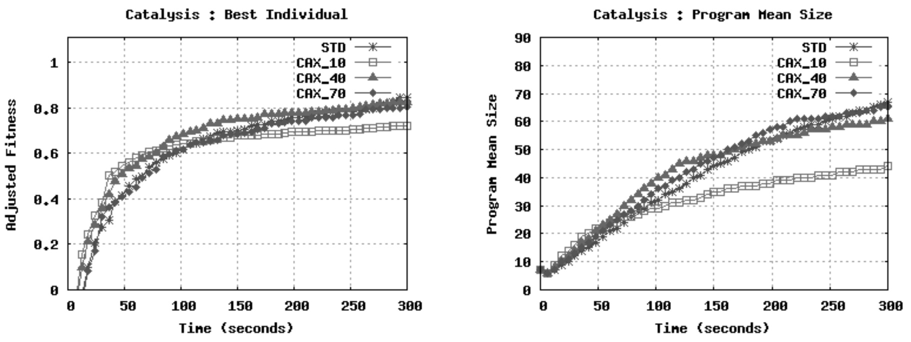


Fig. 5. Results averaged on 4 runs for a reduced population size when CAX starts

Curves were much better although standard GP still got the best result (*cf.* fig. 5 left). In CAX, the individuals size did slightly drop (as claimed in [MR06b]), but not by a large amount, and for individuals that were not as fit as standard GP individuals (*cf.* fig. 5 right).

Note that in [MR06a], fitness curves are given with reference to the *number of generations*, which, with such an operator, is quite meaningless since when the CAX is started, producing one child needs many more evaluations than a standard crossover. The curves of [MR06b] show performance wrt. the *number of evaluations*, which is much better, although still not very accurate, for the reason that in GP, all individuals do not take the same time to evaluate, and according to the paper, CAX leads to smaller individuals that should therefore evaluate faster than individuals produced by a standard GP crossover.

In order to be as precise as possible in the comparison, the presented plots show results against *computing time*, which, in our opinion, is the most accurate metrics (all four plots of this paper are done in parallel, on a quadri-processor exclusively devoted to the runs, to make sure that all time scales are identical).

Moreover, plotting against time also takes into account the potential differences in execution speed induced by the implementation of the different operators.

On this example, it seems that one can conclude that the exhaustive search started by the CAX in order to find the best position for S_2 in P_1 does not yield much better results than when the same amount of CPU time is used by an ordinary standard crossover.

Finally, a two parameter function $kt^n/(1 + kt^n)$ has been extracted from the best Pareto front that was found (using the standard GP operator) that shows the best balance between fitting accuracy and number of parameters. Due to the low number of parameters, overfitting of the neural network has been easily handled, and the resulting architecture shows a very low level of complexity in both the number of hidden layers and total amount of neurons (Multi-layer Perceptron 4:4-6-2:2), four synthesis variables as input, and k and n as output. A new criterion of deactivation has been employed on the resulting virtual curves predicted for the entire synthesis space. Based on the obtained data, a few catalysts (picked up from the best predicted materials) have been synthesized and tested confirming the very high accuracy obtained by such methodology. Consequently, this new strategy used for the first time in heterogeneous catalysis appears to be both very promising and relevant.

4 Re-Implementation of the [MR06b] Benchmarks

These disappointing results asked for more tests concerning the CAX, so it was decided to re-implement the benchmarks of [MR06b], but on a *CPU-time* basis, rather than on an *per-evaluation* basis. This difference is meant, so that the presented results show the efficiency of the CAX under a new point of view (there would be no point in reproducing the experiments presented in [MR06b]).

[MR06b] presented benchmarks on three standard problems taken from Koza's Genetic Programming books [Koz92, Koz94]: the quartic polynomial symbolic regression, the 11 bit multiplexer and the lawnmower with one ADF (Automatically Defined Function). However, since the target problem on catalysts does not need ADFs, it was decided to skip them and implement the quartic polynomial symbolic regression, the 11 bit multiplexer and the artificial ant on the Santa-Fe trail (that does not need ADFs in Koza's implementation).

4.1 Experimental Setup

All benchmarks were re-implemented according to Koza's specifications, and tests on CAX only started when similar results as those presented in [Koz92, Koz94] were obtained.

Where [MR06b] authors specified that they ran all the experiments for 50 generations, and averaged over 50 runs, all the experiments of this paper were done over 50 runs, but for a number of seconds allowing standard GP to perform the same number of evaluations as found in Koza's book.

For the sake of simplicity, the varying rate of crossovers (indicated as `P_var` in [MR06b]), was not re-implemented. The experiments implement the simple solution of turning on the CAX after completion of a certain percentage of a run (which is the solution that gave the best results in [MR06b]). To quote [MR06b]: *if the CAX was turned on after 10% completion of the run, then the first 10% of the run was completed using only standard crossover, and the rest of the run was generated by only using context-aware crossover.*

Here again, the implementations of this paper depart from those of [MR06b] in spite of this precise definition: Majeed and Ryan chose to use a reduced population size (200 individuals) for experiments that used CAX *from the very beginning of the runs*, while in the present paper, in order to precisely evaluate the effects of CAX, *the standard GP population size (4,000) is used in the beginning of CAX runs* until the CAX operator is started, after which the population is reduced by 95% (down to the 200 individuals specified in [MR06b]).

As a consequence, in this paper, the runs using CAX are *identical* to the standard GP run *until the CAX operator is started*, which is not the case in [MR06b]. In other words, in this paper, standard GP is the same as a CAX_100 (i.e. CAX started after 100% of the run).

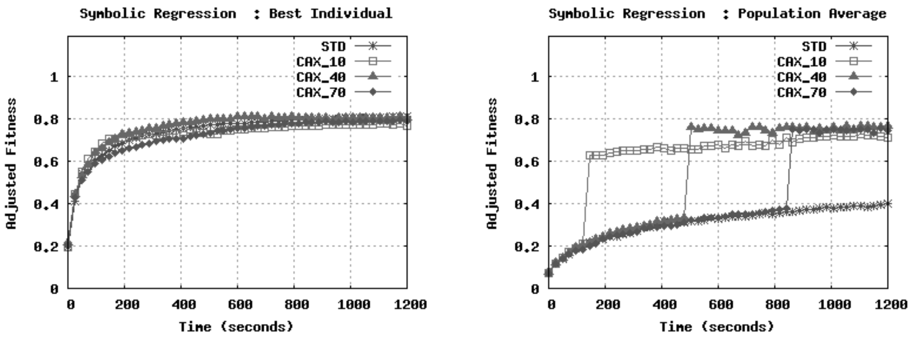


Fig. 6. Quartic polynomial symbolic regression. Left: Best individual performance. Right: Average performance of the population. Standard GP is in fact CAX_100.

To take a precise example, in order to obtain the `CAX_10` curve for the quartic polynomial problem that takes 1200 seconds (*cf.* fig. 6), the algorithm begins with a population of 4,000 individuals for 120 seconds (10% of 1200), after which the CAX is started. At this moment, the population is reduced down to 200 individuals using the following process: the best individual is kept (elitism), and the other 199 individuals are selected with a tournament of size 40 (1% of the original population size). Lower arities were tested, with elitist tournament-7 and random selection, but tournament-40 is what yielded the best results.

4.2 Quartic Polynomial Symbolic Regression Problem

Koza's quartic polynomial symbolic regression problem ($x^4 + x^3 + x^2 + x$) is implemented, with a population of 4,000 until a CAX is started. At this moment, the population is reduced to 200 so as not to freeze the CAX on the same generation till the end of the run.

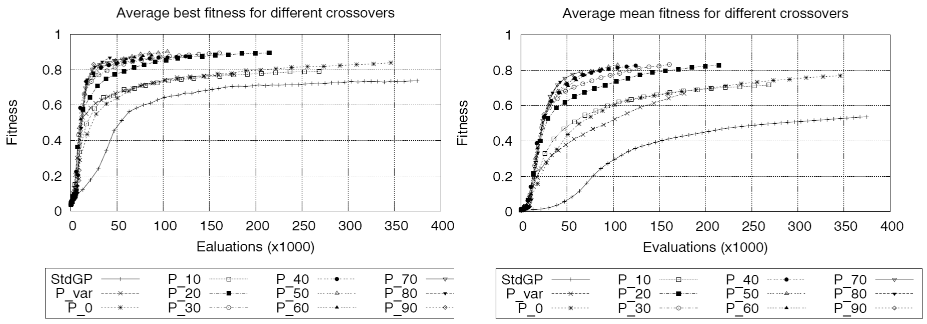


Fig. 7. Figures from [MR06b]. Standard GP is the lowest curve on both figures.

On the Best Individual fitness curve (*cf.* fig. 6 left), surprisingly enough, one can see that all methods perform the same, even when the CAX is started and the population reduced from 4,000 down to 200. However, the Average Population fitness curve (*cf.* fig. 6 right) clearly shows that something is going on.

Whenever the CAX starts, the average population fitness is boosted to values not far from the best individual's, but apparently, this does not lead to premature convergence, which is an interesting feature. Unfortunately, however, the great improvement shown by all CAX best individual curves presented in [MR06b] (*cf.* fig. 7) is not observed. Even if one takes into account the fact that [MR06b] uses only 200 individuals from the beginning on CAX runs, on these curves, the best individual of all CAX runs yields a much better result than the best standard GP individual. We have clearly not been able to reproduce this in our curves.

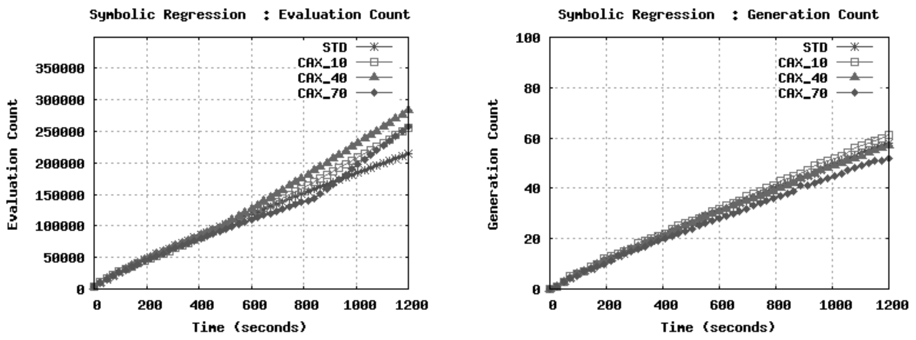


Fig. 8. Evaluation count and generation count

Fig. 8 shows that the implementation population reduction scheme is fair for the CAX evaluation- and generation-wise.

4.3 11 Bit Multiplexer Problem

On this problem, the effects of CAX look pretty much the same: on fig. 9 left, starting the CAX does not seem to have much effect at all (although it seems that

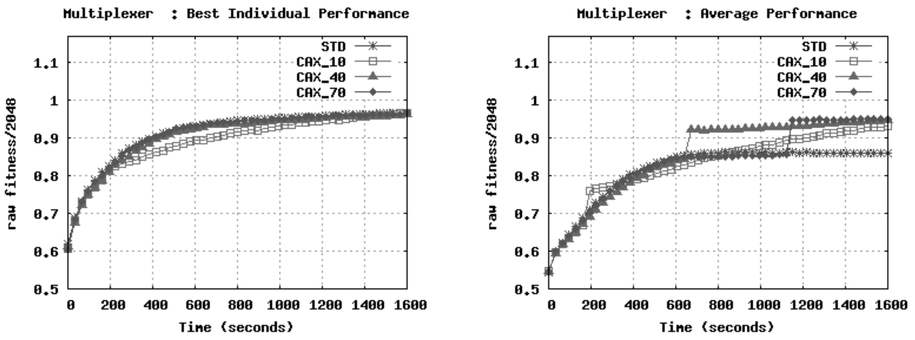


Fig. 9. Best and mean performance on the 11 bit multiplexer problem

CAX_10 has had a small negative impact on the best individual performance). On the right, one can clearly see the effect of CAX on the population average fitness whenever CAX is started. Before CAX starts, the curve is of course identical to standard GP. What is remarkable, though, is that for CAX_10, it seems that the population has not prematurely converged, though the average fitness is very close to the best fitness. In the end, the best individual value for CAX_10 is the same as for standard GP !

4.4 Artificial Ant on the Santa-Fe Trail

The last benchmark in [MR06b] is the Lawnmower problem, as described in [Koz94]. However, this problem uses ADFs that we did not implemented, since the original catalysis problem did not need them. So, in order to take a nevertheless comparable benchmark, the Artificial Ant on the Santa-Fe trail problem (i.e. an implementation without ADFs) was chosen.

On this benchmark, still no improvement on the best fitness (*cf.* fig. 10 left) although this time, CAX_10 does not seem to recover and catch up with

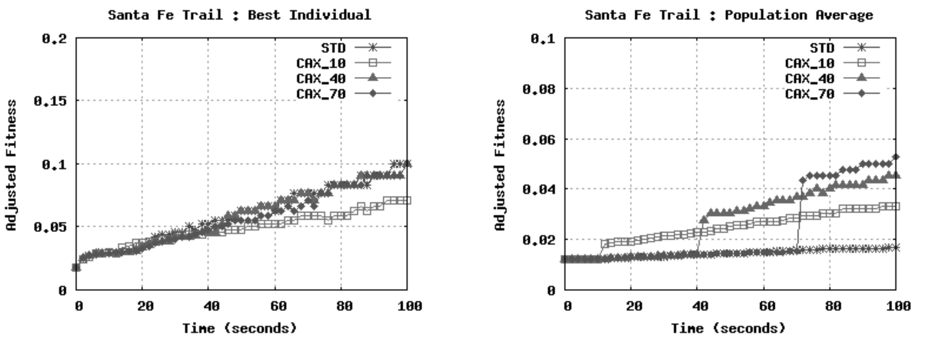


Fig. 10. Artificial Ant on the Santa-Fe Trail. Left: Number of hits of the best individual. Right: Number of hits of the average population.

Standard GP. Here again, a spectacular boost on the population average fitness is observed whenever the CAX starts.

5 Conclusion

The conclusion is not exactly the one that was originally planned. When starting this work, the aim was to improve the best individual result on the heterogeneous catalyst optimisation problem using the CAX. Unfortunately, things did not turn out as expected, as it was impossible to obtain better results with the CAX than with an ordinary crossover operator on this real world problem.

A careful (albeit different) implementation of the benchmarks used in [MR06b] seems to show that Context Aware Crossover is not capable of improving the best fitness value, although CAX seems to be a very good exploitation operator that boosts the whole population towards much better fitness values *while maintaining a good level of diversity* (best individual fitness keeps rising after the CAX is started).

This means that CAX remains a very interesting crossover method, that would deserve another careful investigation on diversity preservation.

References

[AMS+06] Corma, A., Moliner, M., Serra, J.M., Serna, P., Díaz-Cabañas, M.J., Baumes, L.A.: A new mapping/exploration approach for ht synthesis of zeolites. Chemistry of Materials 18, 3287–3296 (2006)

[Bau06] Baumes, L.A.: Map: An iterative experimental design methodology for the optimization of catalytic search space structure modeling. Journal of Combinatorial Chemistry 8, 304 (2006)

[BFLM04] Baumes, L.A., Farruseng, D., Lengliz, M., Mirodatos, C.: Using artificial neural networks to boost high-throughput discovery in heterogeneous catalysis. QSAR and Combinatorial Science 29, 767 (2004)

[BMC07] Baumes, L.A., Moliner, M., Corma, A.: Prediction of itq-21 zeolite phase crystallinity: Parametric versus non-parametric strategies. QSAR and Combinatorial Science 26, 255 (2007)

[CDCJ+06] Corma, A., Jíaz-Cabañas, M., Jordá, J.L., Martínez, C., Moliner, M.: High-throughput synthesis and catalytic properties of a molecular sieve with 18- and 10-member rings. Nature 443, 842–845 (2006)

[KFB+04] Klanner, F., Baumes, L., Mirodatos, C., Schüth, F.: The development of descriptors for solids: Teaching catalytic intuition to a computer. Angewandte Chemie International Ed. 43, 5347 (2004)

[Koz92] Koza, J.R.: Genetic Programming: On the Programming of Computers by means of Natural Evolution. MIT Press, Massachusetts (1992)

[Koz94] Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Massachussetts (1994)

[MR06a] Majeed, H., Ryan, C.: A less destructive, context-aware crossover operator for GP. In: Collet., P., et al. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 36–48. Springer, Heidelberg (2006)

[MR06b] Majeed, H., Ryan, C.: Using context-aware crossover to improve the performance of GP. In: Keijzer, M., et al. (eds.) GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, Seattle, Washington, USA, July 8-12, 2006, vol. 1, pp. 847–854. ACM Press, New York (2006)

[MR07a] Majeed, H., Ryan, C.: Context-aware mutation: a modular, context aware mutation operator for genetic programming. In: Thierens, D., et al. (eds.) GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, London, July 7-11, 2007, vol. 2, pp. 1651–1658. ACM Press, New York (2007)

[MR07b] Majeed, H., Ryan, C.: On the constructiveness of context-aware crossover. In: Thierens, D., et al. (eds.) GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, London, July 7-11, 2007, vol. 2, pp. 1659–1666. ACM Press, New York (2007)

[SBMC08] Serna, P., Baumes, L.A., Moliner, M., Corma, A.: Combining high-throughput experimentation, advanced data modelling and fundamental knowledge to develop catalysts for the epoxidation of large olefins and fatty esters. Journal of Catalysis (in press, 2008)

Countering Poisonous Inputs with Memetic Neuroevolution

Julian Togelius[1], Tom Schaul[1], Jürgen Schmidhuber[1,2], and Faustino Gomez[1]

[1] IDSIA, Galleria 2, 6298 Manno-Lugano, Switzerland
[2] TU Munich, Boltzmannstr. 3, 85748 Garching, München, Germany
{julian,tom,juergen,tino}@idsia.ch

Abstract. Applied to certain problems, neuroevolution frequently gets stuck in local optima with very low fitness; in particular, this is true for some reinforcement learning problems where the input to the controller is a high-dimensional and/or ill-chosen state description. Evidently, some controller inputs are "poisonous", and their inclusion induce such local optima. Previously, we proposed the memetic climber, which evolves neural network topology and weights at different timescales, as a solution to this problem. In this paper, we further explore the memetic climber, and introduce its population-based counterpart: the *memetic ES*. We also explore which types of inputs are poisonous for two different reinforcement learning problems.

1 Introduction

It stands to reason that when applying evolution methods to reinforcement learning problems, providing more information to the controller rather than less should make the problem easier rather than harder to solve. Intuitively, if those parts of the state description that are actually necessary to solve the problem (e.g. the position of the agent relative to the goal) were available, then a sensible learning algorithm ought to disregard any redundant information (e.g. the position of an unrelated agent relative to the goal, a random variable, or relevant aspects of the system state represented in the wrong scale or frame of reference). This assumption is not challenged by most existing reinforcement learning benchmarks, since they provide the controller with only a few well-chosen variables as inputs.

For problems where the best state representation is not immediately obvious, the above assumption is often wrong. In many cases, providing extra information to the controller results in lower fitness. For example, Lucas and Togelius [1] found that removing an input representing an angle to a way point was necessary for successful navigation to evolve for an holonomic agent; Igel [2] found that the CMA algorithm found good pole-balancing controllers much faster when a bias input was removed; and in the domain of helicopter control, De Nardi et al. [3] found that the network controlling yaw and the network controlling the pitch and roll could not share any inputs, lest evolution never found good controllers. In these examples, the presence of certain "poisonous" irrelevant inputs induces local minima in the fitness landscape—evolution exploits the poisonous inputs to quickly find controllers that score better than random but cannot be built upon to find full solutions. For evolutionary reinforcement learning to be useful in real-world problems where the best state description is not known in advance, we need algorithms that can identify those state variables that should be ignored. Such algorithms will likely operate on more than one timescale, with one process learning what to ignore and another process learning the policy.

In a recent paper, we introduced the *memetic climber* [4], a variation of the simple hill-climber that searches for neural network topology and weights at different time scales; each topology mutation is accepted only if it is better than its predecessor after a brief period of local search in weight space. We found that on a version of a simulated car racing task which used a carefully selected set of inputs, the memetic climber performed slightly better than a standard, non-memetic hill-climber. However, when extra, potentially useful, but redundant inputs were added, the standard hill climber failed to find good controllers, while the memetic climbers performed almost as well as with the smaller set on inputs. In other words, the memetic climbers learned which inputs to ignore.

A number of algorithms have been proposed that evolve both topologies and weights of neural networks at the same time (see [5] for an overview). Most of these are not memetic algorithms, and treat topology search and weight search as a single search process, on a single time scale. An exception is the EANT2 algorithm [6], which treats topology and weight search as separate but interdependent processes: it evolves topologies with a simple ES and weights with the CMA-ES. Memetic algorithms have previously been used to efficiently search the space of neural network weights; see [7] for an example.

This paper continues our exploration of when and why extra inputs thwart the learning of effective control policies, and how memetic search in weight and topology space can counter this phenomenon. There are three main objectives: (1) to investigate the effects of changing the number of local search steps per global mutation in the memetic climber, (2) to compare the effects of redundant inputs with and without information content (irrelevant state descriptions versus pure noise), and (3) to introduce a population-based version of the memetic climber, the *memetic ES*, and compare it with other evolutionary algorithms.

2 Neural Memetic Search Algorithms

In this section, we describe five memetic search algorithms for neural network weights and topologies. The first two, originally presented in [4], use a single search point, while the other three are memetic extensions to evolutionary strategies. All of the algorithms are used to search the space of *masked networks*: in addition to the connection weights, the network chromosomes contain a bit-mask with a bit for each connection that determines whether or not the corresponding connection is active in the network.

There are two types of mutation operations that are applied to the masked network representations:

- ***weight mutation*** adds values drawn from a Gaussian distribution to all of weights.
- ***topology mutation*** iterates over all bits in the mask, flipping any bit with probability p.

Algorithm 1. Memetic Climber (n,m)

```
1  INITIALIZE (champion)
2  $f_{champ} \leftarrow$ EVALUATE (champion)
3  for $i=1$ to $n$ do
4      contender $\leftarrow$ champion
5      TOPOLOGYMUTATE (contender)
6      for $j=1$ to $m$ do
7          $f_{cntder} \leftarrow$ EVALUATE (contender)
8          subcontender $\leftarrow$ contender
9          WEIGHTMUTATE (subcontender)
10         $f_{subcnt} \leftarrow$ EVALUATE (subcontender)
11         if $f_{subcnt} >= f_{cntdr}$ then
12             contender $\leftarrow$ subcontender
13         end
14     end
15     $f_{cntder} \leftarrow$ EVALUATE (contender)
16     if $f_{cntder} >= f_{champ}$ then
17         champion $\leftarrow$ contender
18     end
19 end
```

When and how often these two operations are used relative to each other, is the key feature that distinguishes the algorithms presented here.

2.1 Memetic Climber

The memetic climber, described in Algorithm 1, can be considered two nested hillclimbers operating at different timescales, and in different search spaces: a slow search in topology space (the outer loop, lines 3-19), and a fast search in weight space (the inner loop, lines 6-14). The algorithm maintains a single candidate solution, the *champion*. Each “generation”, a copy of the champion, *the contender*, is topology-mutated and then local search is performed in for m steps. The contender replaces the champion only if its fitness after local search is higher than or equal to that of the champion. The intuition behind this algorithm is that by using local weight search to refine new topologies it might be possible to mitigate the disruptive effect of topology mutation. This is related to the NEAT algorithm, which affords new topologies “innovation protection” [8].

2.2 Inverse Memetic Climber

The inverse memetic climber works in the same way as the memetic climber except that the two types of mutation are interchanged (i.e. swapping lines 5 and 9 in Algorithm 1): for every weight mutation, local search is done in topology space.

2.3 Memetic ES

The *memetic ES*, described in Algorithm 2, is one possible combination of the memetic climber and evolution strategies. At each generation, a small amount of

Algorithm 2. Memetic ES(μ,λ,n,m)

```
1  INITIALIZE (Population, $\mu + \lambda$ individuals)
2  for $i$=1 to $n$ do
3      for $j$=1 to $(\mu + \lambda)$ do
4          contender ← COPY (Population[j])
5          $f_{cntndr}$ ← EVALUATE (contender)
6          for $k$=1 to $m$ do
7              subcontender ← contender
8              WEIGHTMUTATE (subcontender)
9              $f_{subcnt}$ ← EVALUATE (subcontender)
10             if $f_{subcnt} >= f_{cntdr}$ then
11                 contender ← subcontender
12             end
13         end
14         Population[j] ← contender
15         EVALUATE (Population[j])
16     end
17     PERMUTE (Population)
18     SORTONFITNESS (Population)
19     for $j$=$\mu$ to $(\mu + \lambda)$ do
20         population ← COPY (Population[j-$\lambda$])
21         TOPOLOGYMUTATE (population[j])
22     end
23 end
```

local search is conducted in weight space for each individual in the population. The population is then sorted by fitness, and the least fit λ are replaced by copies of the better fit μ of the population. Finally, all the newly copied individuals are topology-mutated.

2.4 Inverse Memetic ES

Just as with the inverse memetic climber, the inverse memetic ES is identical to memetic ES except that the weight and topology mutations (lines 8 and 21) are swapped.

2.5 Memetic CMA-ES

This algorithm is a version of the memetic climber where the local search (lines 6-14, algorithm 1) uses *Covariance Matrix Adaption Evolution Strategy* (CMA-ES) instead of the simple hillclimber. CMA-ES is a method that adapts the covariance matrix of the problem variables in order to model the fitness landscape as a multivariate normal distribution that used to generate new search points (see [9] for a complete description of this algorithm). Our Memetic CMA-ES is related to the more complex EANT2 algorithm [6] which uses CMA-ES for weight search and standard ES to search topology.

Table 1. Results for Memetic Climber with different numbers of local, weight search steps. Best fitness found after 20000 episodes for *simplerace*, averaged over 50 runs.

Local steps	Standard	Extra Cartesian	Extra random
2	13.83	12.81	13.27
5	13.79	12.71	13.12
10	**13.85**	**13.06**	**13.33**
25	13.51	12.90	13.15
50	13.86	11.66	12.96
100	13.46	9.50	12.76
250	13.25	7.80	11.74
500	11.19	6.99	10.65

3 Experiments

Two very different domains were chosen as testbeds for the memetic algorithms described above: *simplerace* and *non-markovian double pole balancing*. The standard set of controller inputs normally used in these domains provide sufficient information for the controller to solve the task. In order to evaluate how well the algorithms learn to ignore irrelevant or redundant information, experiments were conducted using three different input representations: (1) the *standard inputs*, (2) an *extended input set* consisting of the standard inputs plus a number of inputs accurately describing redundant or irrelevant aspects of the state, and (3) a *noise input set* consisting of the standard inputs and a number of normally distributed random variables.

To compare the memetic to the non-memetics approaches, experiments we also run for standard (non-memetic) $(5 + 5)$ and $(50 + 50)$ evolution strategies, and as a baseline two versions of random search: one which randomly generates masks and weight vectors, and one which only generates random weight vectors, with all mask bits set. Both versions the memetic and inverse memetic ES have a population size of 10.

The weight mutation for all methods used a Gaussian distribution with mean 0 and standard deviation 0.1 applied to all weights, and the probability of having a bit flipped, p, was set to 0.05 for the topology mutation operator. For the memetic algorithms, all mask bits are initially unset. These two operators are described in section 2.

3.1 Setup: Simulated Race Car Driving

The *simplerace* problem involves driving a car in a simple racing simulation in order to reach as many randomly placed waypoints as possible in a limited amount of time. In addition, the driver must decide which waypoint to target in order to beat an opponent car that tries to reach the same waypoints, giving the game a strategic element. The game has previously been used as a benchmark problem in several papers, and in two competitions associated with recent conferences[1].

[1] A description of the problem is available in [10], and source code can be downloaded from http://julian.togelius.com/cec2007competition.

For this domain, the standard input set consists of eight values: a bias term, the speed of the car, angle and distance to the current and next way point and to the next vehicle. The extended input set consists of the positions of both the controlled and the opponent car in Cartesian space, the speed of the opponent car, and the orientation and angular velocity of the controlled car in Cartesian coordinates. This information, while correct, should be significantly harder than the core inputs to interpret, due to the need for coordinate transformations. The noise input set are seven inputs that are set to independent values drawn from a Gaussian distribution with mean 0 and standard deviation 1.

Each algorithms was run 50 times, and $20,000$ multilayer perceptrons (MLPs) with the *tanh* transfer function and six hidden units were evaluated in each run. For the memetic climber, eight different settings $(2, 5, 10, 25, 50, 100, 250, 500)$ for the number of local search steps, n, (weight mutations) per global (topology mutations) were tried.

3.2 Results: Simulated Race Car Driving

From Table 1, we can see that when the number of local steps per global mutation is low, the memetic climber finds good controllers under all three input conditions; for the *simplerace* problem, the best setting seems to be 10 local search steps, though everything under 50 is good. When using hundreds of search steps, worse solutions are found under all input conditions, though this effect is much more marked under the extra Cartesian input condition.

In Table 2 the results for the best memetic climber configuration are compared with a number of other search algorithms. The most striking result is that for every algorithm, the controllers found using standard inputs are better than those found using the extra random inputs, which in turn are better than those found using the extra Cartesian inputs. These differences can be minor, as for the memetic climber, or drastic, as for the $(50 + 50)$ ES.

Table 2. Results for *simplerace* task. Best fitness found after 20000 episodes for *simplerace*, averaged over 50 runs.

Algorithm	**Standard**	**Extra Cartesian**	**Extra random**
Random search (mask/weights)	13.44	9.08	11.15
Random search (weights only)	10.66	1.18	7.36
Hillclimber	12.82	0.56	10.78
Memetic climber 10-local	13.85	13.06	13.33
Inverse memetic climber 10-local	13.85	12.76	13.52
(5+5) ES	15.47	0.94	13.93
(50+50) ES	**16.23**	1.30	14.14
(5+5) Memetic-ES	15.36	14.02	14.90
(5+5) Inverse Memetic-ES	15.45	**14.51**	**15.18**

None of the algorithms that only search weight space manage to find good controllers using the extra Cartesian inputs, e.g. the $(50 + 50)$ ES finds the best controllers (probably close to the optimum for reactive controllers) for the standard inputs, but performs extremely poorly with the extra Cartesian inputs. This effect can be seen even for random search, where random search in weight space performs worse than random search in topology and weight space under the standard condition, much worse under the extra random condition, and very much worse under the extra Cartesian condition.

Using the standard and extra random random, all the population-based algorithms outperform all non-population-based algorithms. With standard inputs, the ESs slightly outperform the memetic ESs, and under the extra random condition the opposite is the case; however, under the extra Cartesian condition the difference is dramatic. Overall, the memetic ESs are the best algorithms of those compared for finding *simplerace* controllers. (The differences in performance between standard and inverse versions are rather small and unsystematic.)

3.3 Setup: Non-markovian Double Pole Balancing

In this task, two poles, sitting side by side, hinged to a wheeled cart must be balanced simultaneously by applying a scalar force at regular intervals such that they are balanced indefinitely and the cart stays within the track boundaries. Unlike the standard inverted pendulum problem which is nearly linear around the unstable equilibrium point, the double pole system is highly non-linear due to the interacting between the poles. In addition, in this non-Markovian version, the controller only recieves three of the six state variable as standard input: the distance of the cart from the center of the track, and the angle of each pole from vertical. Since the velocity of the cart, and the angular velocities of the poles are not provided to the controller, it must compute them from previous inputs usin internal state (memory) in order to balance the poles (see [11] for equations of motion and system parameters).

For this problem, the extended input set consists of the four Cartesian coordinates of the tips of both poles. The noise input set consist of four Gaussian noise sources as in the *simplerace* setup. In order to compute the velocities, the controllers were represented by Elman-style simple recurrent neural networks with sigmoid transfer functions. Each algorithm was run 30 times, and each run lasted until the best network could balance the pole for $50,000$ time steps, or until $30,000$ networks had been evaluated, whichever occured first. For this problem, we also compare our results to CMA-ES.

3.4 Results: Non-markovian Double Pole Balancing

Table 3 summarizes the results for the pole balancing experiments. The variance in the results of each algorithms for different sets of inputs is more pronounced on this task compared to *simplerace*. In fact, many of the algorithms fail to find controllers that can balance both poles for the required $50,000$ time steps. This is partly because, for run time reasons, we have chosen to cut off the search very early, at $30,000$ evaluations. This is roughly ten times more than the best algorithm needs, but might be too short for some other algorithms.

One result that immediately stands out is that CMA-ES (along with memetic CMA-ES) is much better than all of the other algorithms using both the standard and extra Cartesian inputs. This is to be expected as CMA-ES is one of the

Table 3. Comparison of methods using different input sets for non-Markovian pole balancing task. For each type of input: the first column (evals.) is the average number of pole balancing attempts required to solve the task; the second column (%solved) indicates the percentage of the runs that were able to solve the task; the third column (fail. fit.) is the average final fitness for the unsuccessful runs. The number of local search steps per global mutation is 100 for the memetic climbers, 50 for memetic-ESs and 500 for the memetic CMA-ES. Each method was run 50 times.

Algorithm	**Standard**			**Extra Cartesian**			**Extra random**		
	evals.	%solved	fail. fit.	evals.	%solved	fail. fit.	evals.	%solved	fail. fit.
Random search	—	0	60	—	0	60	—	0	43
Random masks	—	0	56	—	0	60	—	0	49
Hillclimber	—	0	47	—	0	45	—	0	38
(5+5) ES	—	0	46	—	0	43	—	0	39
(50+50) ES	—	0	40	—	0	40	—	0	38
CMA-ES	3658	**90**	**736**	**3421**	**90**	**523**	—	0	92
Memetic CMA-ES	2223	39	497	21733	39	109	**29000**	**5**	162
Memetic climber	**1736**	17	444	8005	2	324	—	0	64
Inverse memetic	2900	2	76	—	0	109	—	0	47
Memetic-ES	1950	5	240	20500	5	147	—	0	**862**
Inv. Memetic-ES	—	—	175	—	0	112	—	0	51

most efficient algorithms for this particular problem to date [11]. However, the performance of CMA-ES drops sharply under the extra random condition–from 90% to 0% successful runs.

In fact, the performance of all algorithms degrades dramatically when the noisy inputs are present. The only algorithms that perform better than random search in this case are the memetic CMA-ES (which sometimes solves the problem) and the memetic ES (which does not solve the problem, but has a relatively good average fitness for failed runs).

Even using the standard and extra Cartesian inputs, the normal memetic algorithms clearly outperform the non-memetic algorithms, including the ES (but not CMA-ES). The inverse memetic algorithms perform worse than the normal memetic algorithms under all conditions.

4 Discussion

Pole balancing and *simplerace* are quite different problems with apparently very different search spaces, for all three input conditions. For the *simplerace* problem, random search does relatively well, and the observed performance differences between algorithms is at the upper end of the fitness range. For pole balancing, almost the opposite is the case: most algorithms spend a long time trying very unfit solutions and some, including the standard ES, do not perform noticeably better than random search within the allotted number of evaluations.

The search spaces of the two problems are also transformed in different ways by the added inputs. The extra Cartesian inputs make the *simplerace* problem hard to solve for algorithms that search only for weights, but have little effect on pole balancing. This is possibly due to the fact that the Cartesian inputs make the *simplerace* search space deceptive: algorithms easily find modestly fit solutions that depend on the extra inputs, but are far from truly solving the task. No such effect seems to exist for the extra Cartesian inputs for pole balancing.

Adding random extra inputs has little effect on the *simplerace* problem, while it completely transforms the pole balancing problem, making the search much harder for all tested algorithms. A simple explanation for this is that noise is more deleterious for an unstable system like the double pole balancer, so that the connections from the random inputs have to be masked off or made very close to zero in order to prevent them from perturbing the system. The *simplerace* environment, however, does not exhibit this kind of instability, so that steering the car is more robust to disturbances (i.e. each action can, in principle, correct for each disturbance without the system diverging).

For both problems, memetic search of topologies and weights was not only competitive with algorithms that searched only for weights using standard inputs, but also significantly outperformed them when extra “distracting” inputs were provided.

The decrease in evolvability in the presence of poisonous inputs is not due to the increased dimensionality of the input or search space. This is clear from the fact that, for both problems, evolvability decreased only very slightly in the presence of non-poisonous inputs, whereas the input dimensionality is identical. Further experiments (omitted due to space constraints) have shown that the effects of poisonous inputs persist when replacing MLPs with linear filters of much lower dimensionality. See also the experiments in chapter 7.2 of [10], which suggest that increasing the dimensionality of the search space (through increasing the size of the hidden layer) actually *increases* evolvability in *simplerace.*

In *simplerace*, random weight and topology search reached lower fitness when subjected to the noise inputs than under the other two conditions, but performance did not decrease as drastically as it did for the other search algorithms. This suggests that poisonous inputs decrease evolvability both by reducing the portion of the search space which contains good solutions, and by making the search space more deceptive. The poor performance of random search on pole-balancing prevents us from drawing similar conclusions for that domain.

It is plausible that other topology and weight-evolving algorithms are equally capable of countering poisonous inputs. If so, this provides an important argument for the use of such algorithms on problems where the best input representation is not yet known.

5 Conclusions

We have shown that two rather different control learning problems can be made much harder by adding irrelevant information to the controller inputs, albeit which sorts of extra inputs were poisonous depended on the problem. This effect seems to be independent of the type of neural network used. We have also shown that these effects can be mitigated to a large extent by using algorithms that search topology and weight space separately; such algorithms could solve both benchmark problems in the presence of irrelevant inputs that made them unsolvable by the tested non-memetic algorithms. In particular, we have shown that the memetic climber can be extended to a population-based algorithm in the form of the memetic ES. This algorithm is competitive with other population-based algorithms even when the state description is well-selected and well-represented, and performs better than the memetic climber when presented with problematic irrelevant inputs. We expect these findings to be highly relevant for cases where evolutionary reinforcement learning is applied to novel, unanalyzed problems, for which the correct state description is unknown, so that the best approach is to feed the controller any information that might or might not be relevant. In other words, the sort of problems where you would expect evolutionary computation to be at its greatest advantage.

Acknowledgments

This research was supported in part by the Swiss National Science Foundation (SNF) grant number 200021-113364/1. Thanks to Dan Ashlock for suggesting the term ”poisonous inputs.”

References

1. Lucas, S.M., Togelius, J.: Point-to-point car racing: an initial study of evolution versus temporal difference learning. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2007)
2. Igel, C.: Neuroevolution for reinforcement learning using evolution strategies. In: Proceedings of the Congress on Evolutionary Computation (CEC) (2003)
3. De Nardi, R., Togelius, J., Holland, O., Lucas, S.M.: Evolution of neural networks for helicopter control: Why modularity matters. In: Proceedings of the IEEE Congress on Evolutionary Computation (2006)
4. Togelius, J., Gomez, F., Schmidhuber, J.: Learning what to ignore: memetic climbing in weight and topology space. In: Congress on Evolutionary Computation (CEC) (to be presented, 2008)
5. Yao, X.: Evolving artificial neural networks. Proceedings 1447, 87(9) (1999)
6. Siebel, N.T., Sommer, G.: Evolutionary reinforcement learning of artificial neural networks. International Journal of Hybrid Intelligent Systems 4(3), 171–183 (2007)
7. Krasnogor, N., Pacheco, A.A.,, J.: Memetic algorithms. In: Metaheuristics in Neural Networks Learning, pp. 225–247. Springer, Heidelberg (2006)
8. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
9. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
10. Togelius, J.: Optimization, Imitation and Innovation: Computational Intelligence and Games. PhD thesis, Department of Computing and Electronic Systems, University of Essex, Colchester, UK (2007)
11. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Accelerated neural evolution through cooperatively coevolved synapses. Journal of Machine Learning Research 9, 937–965 (2008)

Parameter Control Methods for Selection Operators in Genetic Algorithms

P. Vajda, A.E. Eiben*, and W. Hordijk

Vrije Universiteit Amsterdam
gusz@cs.vu.nl
http://www.cs.vu.nl

Abstract. Parameter control is still one of the main challenges in evolutionary computation. This paper is concerned with controlling selection operators on-the-fly. We perform an experimental comparison of such methods on three groups of test functions and conclude that varying selection pressure during a GA run often yields performance benefits, and therefore is a recommended option for designers and users of evolutionary algorithms.

1 Introduction

Evolutionary computing (EC) has become a proven problem solving technology over the past few decades [6]. However, the performance of *evolutionary algorithms* (EAs) depends largely on their parameters, such as population size, selection pressure, crossover and mutation rates. Choosing good values for EA parameters before an EA run (parameter tuning) and/or appropriately varying parameter values during an EA run (parameter control) is still one of the main challenges of the field [4]. The ultimate goal is to develop methods that are capable of adjusting the parameter values to a given problem, and also to the different stages of the search process. Traditionally, most attention has been paid to parameters of the variation operators (in particular the crossover and mutation rates) and –to a much lesser extent– to population size [3,13]. In contrast, on-the-fly control of selection operators has received little attention, and the few available results are scattered throughout the literature. Note that in EAs that use evolution strategy-like population management, that is, (μ, λ) or $(\mu+\lambda)$, the offspring population size λ determines the selection pressure, and can thus be used to control the selection operator [11]. However, in this study we focus on *genetic algorithms* (GAs), in particular steady-state GAs that inherently feature two selection mechanisms: one for parent selection and one for survivor selection (replacement). To keep things simple, we keep the replacement strategy fixed and focus on parent selection. The objectives of this paper are twofold: To present an overview of known parameter control methods for (parent) selection operators, and to perform an experimental comparison of these methods with each other and a benchmark GA (a simple GA with K-tournament selection).

* Corresponding author.

To broaden the support base of our empirical findings we use three groups of test functions: (1) a collection of popular test functions for EA research, (2) known "GA teasers", and (3) a set of abstract fitness landscapes created by a randomized generator. The results show that the winning policy[1] largely depends on the group of test functions. Consequently, the overall winner depends on the relative weights these groups are given when calculating the final score for each policy. However, the data provides sufficient support for the superiority of on-the-fly control of selection pressure (as opposed to keeping it constant during a GA run) and an indication of a control mechanism that is capable of regulating selection pressure by itself.

2 Parameter Control for Selection Operators

In this section we present the contestants of our experimental comparison, including K-tournament selection with a fixed K (benchmark), K-tournament selection with changing values of K (new method, introduced here), and three control methods from the literature. Despite an extensive literature study we could not find any other existing control methods, except variations on the Boltzmann selection, e.g., [14].

Fixed tournament size. As a benchmark for the parameter control methods, we use a *simple genetic algorithm* (SGA) with tournament selection with fixed tournament size K. We consider 10 values, $K = 1, 2, 4, 6, 8, 12, 16, 24, 32, 64$, for a comparison with the control methods, where $K = 1$ amounts to uniform random selection (compensated by the replacement strategy).

Deterministic tournament-size control. This method (DTC) is the most straightforward way of adjusting the selection pressure during the search. The tournament size K is a deterministic function of the time step (generation) t:

$$K(t) = \begin{cases} \frac{t(p2-p1)}{1000} + p1 & \text{if } t \in [0, 1000] \\ p2 & \text{otherwise} \end{cases} \tag{1}$$

where p_1 and p_2 are parameters of the method. In words, the tournament size increases linearly from p_1 to p_2 (or decreases if $p_1 > p_2$) for the first 1000 generations, after which it stays fixed at p_2.

Boltzmann selection with annealing. A more sophisticated deterministic parameter control method is Boltzmann selection with a Riemann-Zeta annealing schedule (BSR). The probability $p(x,t)$ of selecting an individual x from the population P_t at time step t is calculated as:

$$p(x,t) = \frac{e^{\gamma_t \cdot f(x)}}{\sum_{y \in P_t} e^{\gamma_t \cdot f(y)}}, \quad \gamma_t = \gamma_0 \sum_{k=1}^{t} \frac{1}{k^\alpha} \tag{2}$$

where γ_t is the annealing temperature. γ_0 and α are parameters of the method.

[1] By policy we mean either a control mechanism, or the value of K in the SGA using K-tournament selection.

Self-adaptive tournament size. It is also possible to let the parameter value be controlled by the evolutionary process itself. In [5], a self-adaptive tournament size (SAT) method was introduced, where an extra parameter $k \in (0,1)$ is added to each individual's chromosome, which represents this individual's contribution to the overall tournament size parameter $K = \left\lceil \sum_{i=1}^{N} k_i \right\rceil$, where N is the population size. Note that $K \in [1, N]$. The self-adaptive mechanism for mutation rates in GAs as described in [2] is then used to mutate the individual values of k:

$$k' = \left(1 + \frac{1-k}{k} \cdot e^{-\gamma \cdot N(0,1)}\right)^{-1} \tag{3}$$

where γ is a learning rate which allows for control of the adaptation speed. This mutation mechanism has the desirable property that if $k \in (0,1)$, then also $k' \in (0,1)$. A variant of this, called *hybrid* self-adaptive tournament size (HSAT; see [5] for details) adjusts an individual's parameter value k according to whether its fitness value is better or worse than that of its parent:

$$k' = \begin{cases} (1 + \frac{1-k}{k} e^{-|\gamma N(0,1)|})^{-1} & \text{if } f(x') \geq f(x) \\ (1 + \frac{1-k}{k} e^{|\gamma N(0,1)|})^{-1} & \text{otherwise} \end{cases} \tag{4}$$

where x represents the parent and x' the offspring. Of course the $\geq$ sign will be a $\leq$ for minimization problems. Again, γ is a learning rate. This method is self-adaptive as in the original SAT method, but it also uses feedback from the search (in particular the fitness values of parents and offspring).

Fuzzy tournament selection. This method (FTS) is based on the adaptation of selection in [8]. The concept of fuzzy logic (FL) itself was conceived by Zadeh [7]. Here, we use two inputs, genotypic and phenotypic diversity, and one output, μ, the modification parameter. *Genotypic diversity* (GD) is calculated as follows:

$$GD = \frac{\overline{d} - d_{min}}{d_{max} - d_{min}} = \frac{\frac{1}{N}\sum_{i=1}^{N} D(C_{best}, C_i) - min\{D(C_{best}, C_i)\}}{max\{D(C_{best}, C_i)\} - min\{D(C_{best}, C_i)\}} \tag{5}$$

where N is the population size, $D(x,y) \in [0,1]$ is the (normalized) Hamming distance of genomes x and y, and C_{best} is the chromosome of the best individual in the population. *Phenotypic diversity* (PD) is calculated as follows:

$$PD = \begin{cases} \frac{f_{best}}{\overline{f}} & \downarrow \text{for minimization problems} \\ \frac{1-f_{best}}{1-\overline{f}} & \uparrow \text{for maximization problems} \end{cases} \tag{6}$$

The modification parameter μ is calculated using the fuzzy logic rules shown in figure 1. The tournament size K in generation t is set to $K_t = (\mu + 0.5) \cdot K_{t-1}$. The label set for the output is {Low, Medium, High} (figure 1), calculated as follows:

GD \ PD	*Low*	*Medium*	*High*
Low	Low	Medium	High
Medium	Low	High	High
High	Low	Low	High

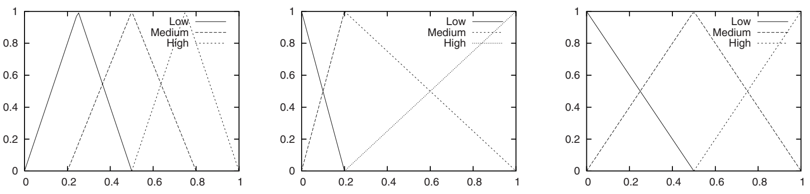


Fig. 1. Fuzzy sets of GD (left), PD (center), and μ (right)

3 Test Suite

To compare the different parameter control methods for selection operators, we have composed an extensive test suite that includes (1) some popular (mostly difficult) functions such as Ackley's function and Rastrigin's function, (2) well known "GA teasers" such as the long path problem and the royal road function, and (3) a set of instances from the multimodal problem generator of Spears. In this section, a brief review of the used test functions is presented.

Popular test functions
These functions are used very often in experimental EC research. Therefore, we restrict ourselves to simply listing them: the Sphere, Generalized Rosenbrock, Generalized Ackley, Rastrigin, and Griewank functions [16,1]. These are all defined as minimization problems on real numbers in L dimensions.

GA teasers
These functions where either specifically constructed to be deceptive or to test certain assumptions about how the genetic algorithm performs its search. They are all defined as maximization problems on bit strings of length L.

Long path. This function [9] is difficult mainly because (at least from a hill-climbing point of view) the only way to the global optimum is very long and narrow, with the length of this path increasing exponentially with the problem size L. So, even though this problem is unimodal, it may take an exponentially long time to reach the optimum.

Ugly. This function [18] presents a deceptive function. To calculate an individual's overall fitness value F, first its chromosome is cut into 3-bit substrings $b_i b_{i+1} b_{i+2}$, and each of these substrings is assigned a fitness value f. The overall fitness of an individual is the average over all the substrings:

$$F(\boldsymbol{b}) = \frac{\sum_{i=0}^{L/3-1} f(b_{3i+1}b_{3i+2}b_{3i+3})}{L}, \quad f(\boldsymbol{b}) = \begin{cases} 2 & \text{if } \boldsymbol{b} \in \{0**\} \\ 3 & \text{if } \boldsymbol{b} = 111 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Royal road. This function [15] was originally introduced to investigate the validity of the building block hypothesis [10]. The main idea of the function is to compare a given individual with a set of schemata with a varied number of defined bits and containing various pre-defined building blocks. For each of the schemata that the individual is an instance of, it receives one point. The overall fitness value of the individual is then a weighted sum of the received points.

Random landscapes by the multimodal problem generator
A useful and tunable test suite for testing GAs can be created with the *multimodal problem generator* (MPG) of Spears [12,17]. Here, we generate problem instances with P = 1, 2, 5, 10, 25, 50, 100, 250, 500, and 1000 peaks whose heights are linearly distributed and where the lowest peak has height 0.5. The fitness of an individual is measured by the Hamming distance (HD) between the individual and the nearest peak, scaled by the height of that peak. The nearest peak is determined by

$$Peak_{near}(\boldsymbol{b}) = \min_{i=1}^{P} (HD(\boldsymbol{b}, Peak_i)) \tag{8}$$

In case of multiple peaks at the same (minimum) distance, the highest of these peaks is chosen. The evaluation function of an individual is then:

$$F(\boldsymbol{b}) = \frac{L - HD(\boldsymbol{b}, Peak_{near}(\boldsymbol{b}))}{L} \cdot height(Peak_{near}(\boldsymbol{b})) \tag{9}$$

4 Experimental Setup

We used all parameter control methods for selection described in section 2 on all test functions in the test suite described in section 3. For each method, 100 runs are performed on each problem instance (with different random seeds for each run), for a maximum of 5000 fitness evaluations per run. Three standard performance measures are used: *Success Rate* (SR; fraction of times the global optimum was found), *Average number of Evaluations to Solution* (AES; only calculated over those runs where the global optimum was found), and *Mean Best Fitness* (MBF). These performance measures are calculated over the 100 runs for each combination of control method and problem instance. We used two variants of the simple genetic algorithm, one with a floating point representation and one with a bit string representation.

For the popular test functions we use a floating point GA, cf. table 1. Note that all problems in this group are minimization problems in $L = 30$ dimensions/variables. Float vector mutation works as follows. Individuals are represented as $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_L \rangle$. The mutation rate $p \in [0, 1]$ determines whether a particular value x_i will be mutated. If so, then x_i' is drawn from a uniform random distribution from the interval $I = [x_i - r, x_i + r]$, where r is defined as:

$$r = \begin{cases} p\frac{b-a}{2} & \text{if } a \leq x_i - p\frac{b-a}{2} \text{ and } b \geq x_i + p\frac{b-a}{2} \\ x_i - a & \text{if } a > x_i - p\frac{b-a}{2} \\ b - x_i & \text{if } b < x_i + p\frac{b-a}{2} \end{cases} \tag{10}$$

where the domain of the chromosome is $[a, b]^L$. The used domain values for the different functions are $[-5.12, 5.12]$ for the Sphere, Rosenbrock, and Rastrigin functions, $[-32.768, 32.768]$ for the Ackley function, and $[-600, 600]$ for the Griewank function.

Table 1. Details of the GAs used in the experiments

Representation	floating point	bit string
GA model	steady-state	steady-state
Optimization	minimization	maximization
Chromosome length (L)	30	29, 30, 64, 100
Population size	100	100
Selection	all from section 2	all from section 2
Crossover	uniform ($p_c = 0.7$)	uniform ($p_c = 0.7$)
Mutation	float vector ($p_m = 0.1$)	bit flip ($p_m = 0.01$)
Replacement	delete worst	delete worst

Since the GA teasers and the multimodal problems are defined in terms of bit strings, we use a GA with a bit-string representation here, see Table 1, right column. The chromosome lengths depend on the individual problem, and here we use $L = 29$ for the long path, $L = 30$ for the ugly problems, $L = 64$ for the royal road problem (with the "building blocks" being of length 8), and $L = 100$ for the multimodal problem instances. The parameter values come from the definition of the functions and most relevant articles. If there were no sufficient and accepted parameter values in the literature (BSR, SGA and DTC), we tried to cover the parameter space by using multiple values.

Finally, the parameter values for the various control methods need to be specified. For the deterministic tournament-size control (DTC) method, we use $p_1, p_2 \in \{2, 7, 17\}$, i.e., six combinations of p_1 and p_2 values such that $p_1 \neq p_2$. For the Boltzmann selection (BSR) method, we use the following values for the (γ_0, α) parameter combination: (40.89, 1.0001), (60.57, 1.1), (154.30, 1.5), and (243.2, 2). And for the self-adaptive tournament size (SAT) and the hybrid version (HSAT), we use $\gamma = 0.22$.

5 Results

For a direct comparison, we calculated a normalized score for each method. On each test function, the worst method gets a score of 0, the best one gets 1, and the remaining ones get a score that is linearly dependent on their result ($RES_{A,T}$):

$$score(A, T) = \begin{cases} \frac{RES_{A,T} - MIN(RES_{.,T})}{MAX(RES_{.,T}) - MIN(RES_{.,T})} & \text{if } \uparrow \\ \frac{MAX(RES_{.,T}) - RES_{A,T}}{MAX(RES_{.,T}) - MIN(RES_{.,T})} & \text{if } \downarrow \end{cases} \tag{11}$$

where A is a selection method and T a test function. Recall that there are three groups of test funtions (*Popular, Teasers,* and *MPG*). Each method gets a score between 0 and 1 for each group by averaging the scores over all test functions in the group. The total score of a method is the average of its scores for all three groups. This total score is calculated for each of the performance measures (*SR, AES, MBF*). The average of these three scores is then the final score of a method (Table 2, last column). Of course we can also calculate a total score for each performance measure first, and then calculate the final score as an average over these three scores. Table 2 shows the results of both these calculations, i.e., either by performance measure (columns 2–4) or by test function group (columns 5–7), both leading to the same final score (last column). As the table shows, the best overall score is achieved by the HSAT method.

Table 2. Scores of selection methods by the three performance measures and by the three groups of test functions. The final score is the average of these.

	SR	AES	MBF	Teasers	Popular	MPG	Final score
HSAT	0.832	0.764	0.828	0.780	**0.977**	0.667	**0.808**
SGA 1	**0.949**	0.568	**0.840**	**0.889**	0.869	0.599	0.786
DTC 2 7	0.720	0.863	0.737	0.682	0.958	0.679	0.773
SGA 2	0.822	0.695	0.748	0.715	0.918	0.632	0.755
SAT	0.752	0.810	0.684	0.692	0.927	0.627	0.749
DTC 7 17	0.712	0.875	0.631	0.548	0.956	0.713	0.739
SGA 4	0.734	0.824	0.657	0.625	0.922	0.668	0.739
DTC 2 17	0.752	0.751	0.689	0.501	0.967	0.724	0.731
FTS	0.714	0.870	0.609	0.594	0.919	0.682	0.731
SGA 8	0.755	0.808	0.619	0.498	0.934	0.751	0.727
DTC 7 2	0.760	0.852	0.549	0.531	0.874	**0.756**	0.720
SGA 6	0.667	0.852	0.590	0.541	0.937	0.630	0.703
SGA 16	0.648	0.860	0.589	0.507	0.929	0.661	0.699
SGA 12	0.681	0.860	0.541	0.470	0.930	0.682	0.694
SGA 24	0.630	**0.886**	0.458	0.409	0.909	0.655	0.658
BRS 2	0.653	0.880	0.441	0.402	0.901	0.671	0.658
SGA 64	0.661	**0.886**	0.417	0.437	0.838	0.690	0.655
DTC 17 2	0.748	0.769	0.434	0.504	0.737	0.710	0.650
SGA 32	0.618	0.879	0.395	0.413	0.813	0.665	0.631
DTC 17 7	0.666	0.759	0.380	0.370	0.784	0.653	0.602

Assigning equal weights to each group of test functions or performance measures is rather arbitrary, and obviously the overall winner could change if we use different weights (which would reflect different relative importances for the different types of test functions or performance measures). In Figure 2 we present the overall winner for all possible combinations of weights. In the three corners of each graph either the three test function groups (left) or the three performance measures (right) are shown, representing a weight of 1 for the group belonging to this corner and 0 for the other two. The equally-weighted average (as given in table 2) is the point in the center. As the figure shows, HSAT remains the best method for a wide range of relative weights (in both cases). The SGA 1 method also performs quite well, but mostly on the GA teasers group, which represents the most difficult test functions. On these functions, a low selection pressure works best (allowing for more exploration), and with the SGA 1 method there is no selection pressure at all (parents are chosen at random with uniform probability). As far as we know, Figure 2 introduces a new way of presenting

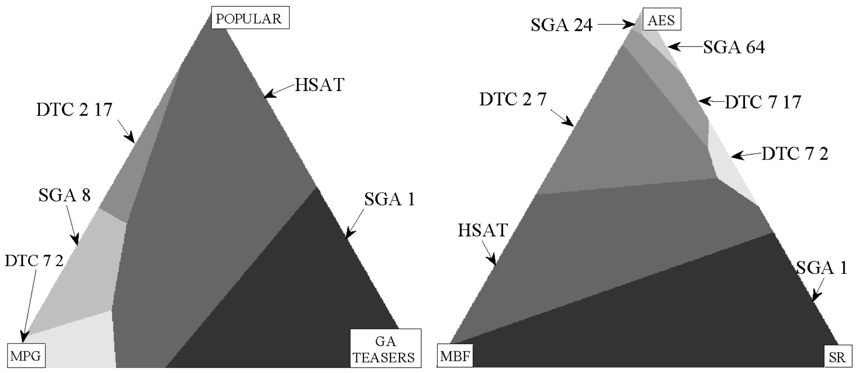


Fig. 2. The best selection methods for different weights of the test function groups (left) and of the performance measures (right). The equally-weighted average is the point in the center.

EA performance results which provides useful insights for algorithm comparisons relative to importance given to different test functions or performance measures.

Given these results, we are now able to answer the question on which test functions does each selection method perform well, and why?

SGA. This is the benchmark for the parameter control methods. Our results show that the optimal tournament size depends strongly on the test function. There are some easy landscapes where the GAs with higher selection pressure perform better (*MPG*), and there are difficult landscapes where the SGA behaves in the opposite way (*GA teasers*). However, the optimal tournament size can be anywhere in between, so we can conclude that there is no prefect (constant) tournament size.

DTC. The results show clearly that there exist landscapes where DTC performs better than the SGA. On the *Ackley* function, e.g., the gradual increase in tournament size leads to a much better performance than all of the SGA variants. This confirms that the optimal tournament size indeed requires adaptation during the runtime of the GA.

BRS. This proves to be a good method on *MPG* problems. However, since it uses a deterministic parameter control, it does not work well on more difficult problems. Due to the parameter settings, the BRS method reaches a very high selection pressure when the population is close to the optimum, which makes the method very fast. Nevertheless, if the landscape does not need a high selection pressure, this method does not give satisfactory results.

HSAT. Overall, this method performs well on all test functions. For the Spears functions, the control parameter (K) increases very much at first, and then drops down again when the population is very close to a peak. This is due to the fact that recombination can initially create better offspring quite easily, so a higher selection pressure converges more quickly to a good area in the landscape, whereas once it becomes more difficult to find better offspring, a lower selection pressure could be beneficial (allowing for more exploration). On more difficult problems, like the GA teasers, the parameter value stays low all throughout, because it is always difficult to find fitter offspring, and a low selection pressure (more exploration) is better – as the good score of SGA 1 indicates. Whenever a landscape requires a high selection pressure, the control parameter always reaches the upper limit (e.g., on most popular functions).

FTS. Although this method is not performing as well as the others, its adaptation is handled quite easily. The fuzzy system does actually not directly control the selection parameter, but it merely multiplies it. Therefore, it usually becomes too high, and thus the selection parameter can not converge to an optimal value.

Finally, we can answer similar questions about the test functions: How "difficult" is a test function and which methods perform well on it?

Popular functions. This group of functions has a variety of difficulty levels, but are overall easier than the GA-teasers. Our results suggest that those methods which can adjust the selection pressure quickly work better. The HSAT method appears to perform the best overall.

GA-teasers. These are the most difficult test functions. Generally speaking, the methods that maintain a low selection pressure perform better, because the population can leave the (bad) local minima more easily. The SGA 1 method appears to be the best in this respect, with HSAT being second.

MPG. Because of the special properties of these functions, the greedy methods found the optimum faster. A high selection pressure gives an advantage on these functions. Here, the DTC (7, 2) method is the best.

6 Conclusions

The main conclusions that can be drawn from our comparison of the different selection methods are as follows:

1) On-the-fly adjustment of the parameter(s) regulating selection is superior to using a constant value. First, because this enables the GA to use different levels of selection pressure on different landscapes without human tuning. Second, because using different parameter values at different stages of the GA search can significantly improve the performance.
2) The best method to control selection depends on the problem (type), but our results indicate that HSAT is the generally best policy. It is capable of appropriately adjusting the selection pressure according to the search results so far *and* to the given landscape.
3) The general strategy of the HSAT method is that as long as improvements are found, the selection pressure will increase to exploit good solutions. However, when the population is stuck or has converged too much, the selection pressure

will decrease to encourage more exploration. These parameter adjustment decisions can be made based on genotypic and/or phenotypic diversity, or from the fitness differences between parents and offspring. Here we only tested one possible implementation of the main idea.

In the future, more detailed investigations will have to be done, but our experiments have already provided useful results and a proof-of-principle. In general, the use of parameter control for selection operators is a recommendable design heuristic for EA users.

References

1. Bäck, T.: Evolutionary algorithms in theory and practice. Oxford University Press, Oxford (1996)
2. Bäck, T., Schütz, M.: Intelligent mutation rate control in canonical genetic algorithms. In: Michalewicz, M., Raś, Z.W. (eds.) ISMIS 1996. LNCS, vol. 1079, pp. 158–167. Springer, Heidelberg (1996)
3. DeJong, K.: Parameter setting in EAs: a 30 year perspective. In: Parameter Setting in Evolutionary Algorithms, pp. 1–18. Springer, Heidelberg (2007)
4. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
5. Eiben, A.E., Schut, M.C., de Wilde, A.R.: Boosting genetic algorithms with self-adaptive selection. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1584–1589 (2006)
6. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, Corrected reprint. Springer, Heidelberg (2007)
7. Yager, R.R., et al.: Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh. John Wiley, New York (1987)
8. Herrera, F., Lozano, M.: Fuzzy genetic algorithms: issues and models. Technical report, No. 18071, Granada, Spain (1999)
9. Hohn, C., Reeves, C.: Are long path problems hard for genetic algorithms? In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 134–143. Springer, Heidelberg (1996)
10. Holland, J.H.: Adaption in Natural and Artificial Systems. University of Michigan Press (1975)
11. Jansen, T., De Jong, K., Wegener, I.: On the choice of offspring population size in evolutionary algorithms. Evolutionary Computation 13(4), 413–440 (2005)
12. De Jong, K.A., Spears, W.M.: A formal analysis of the role of multi-point crossover in genetic algorithms. Annals of Mathematics and Artificial Intelligence (5), 1–26 (1992)
13. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol. 54. Springer, Heidelberg (2007)
14. Mahfoud, S.W., Goldberg, D.E.: Parallel recombinative simulated annealing: a genetic algorithm. Parallel Computing 21(1), 1–28 (1995)
15. Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and GA performance. In: Varela, F.J., Bourgine, P. (eds.) Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, Paris, 11–13, 1992, pp. 245–254. A Bradford book, The MIT Press (1992)

16. Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley, Chichester (1995)
17. Spears, W.: Evolutionary algorithms: the role of mutation and recombination. Springer, Heidelberg (2000)
18. Whitley, D.: Fundamental principles of deception. In: Morgan Kaufmann (ed.) Foundations of Genetic Algorithms, pp. 221–241. Morgan Kaufmann, San Francisco (1991)

Evaluation and Diversity in Co-evolution

Rients P.T. van Wijngaarden and Edwin D. de Jong

Universiteit Utrecht
Algorithmic Data Analysis Group
The Netherlands
wijngaar@phil.uu.nl,
dejong@cs.uu.nl

Abstract. This paper studies the performance of four alternative evaluation methods; two instances of the Exponential Moving average, the Elo-rating and the Glicko-rating method. These methods are tested in a co-evolutionary setup using the LINT-game, which is known to be problematic under co-evolutionary conditions. Besides the different evaluation approaches, two methods aimed at preserving diversity are tested. By using the Objective Fitness Correlation as an analytical tool for monitoring accuracy of evaluation, it is shown that actual performance of an evaluation method strongly depends on whether co-evolutionary failure occurs and that a multi-modal approach to the LINT-problem is effective in maintaining stable progress over time.

1 Introduction

Co-evolution offers the potential to evaluate individuals using a limited, adaptive set of interaction partners. The design of a co-evolutionary algorithm begins with a consideration of the desired *solution concept* [1]. A solution concept specifies which elements of the search space qualify as solution to a given problem and which do not. Examples of solution concepts include: Maximum Expected Utility (MEU), the Pareto-optimal set and Nash-equilibria. In this paper the focus is on *test-based* problems where candidate solutions are evaluated on the outcome against a set of test conditions and the MEU solution concept is thus understood as maximizing the sum of outcomes against all possible opponents.

In the co-evolutionary algorithm the set of test conditions can be understood as a population in its own right with its own evolutionary dynamics, giving us two conceptually distinct populations. The advantage of the evolutionary interaction between the set of candidate solutions and its set of opponents is twofold: biases and overfitting related to the use of a static test set are avoided, while the evolutionary co-dependence of the sets explores the problem space in a more meaningful way than a series of stochastically assembled test sets could.

The co-evolutionary interaction also leads to a set of pathologies that can lead to co-evolutionary failure and threaten stable progress towards the solution concept. Three of these problems recognized by many researchers are *disengagement*, *overspecialization* and *cycling*, see among others [2]. These pathologies can be directly ascribed to a dissociation between the *subjective* fitness, derived from the local interactions with other individuals, and the *objective* fitness, a measure of individual performance on a global scale.

Despite these problems co-evolutionary algorithms have been designed that guarantee stable progress for the MEU solution concept and the other solution concepts described above. The practical value of such guarantees remains limited however, as long as bounds on the computational expenses required to actually reach the desired solution concept are unavailable. A central current challenge in co-evolutionary research therefore is how algorithms can be designed that not only enable stable progress, but also do so *efficiently*, i.e. using limited computational resources.

In this paper four alternative evaluation methods are compared to a control method in terms of the accuracy of evaluation. This accuracy is measured by means of the Objective Fitness Correlation (OFC) [3], an analytical tool that expresses whether the evaluation of a population adequately reflects the global goal. Furthermore two alternative algorithmic setups are used to test the evaluation methods that differ in the way diversity within populations is handled to see if the performance of the evaluation methods can be viewed in isolation. A test-problem is used that is known to lead to co-evolutionary failure and is yet simple enough to study this failure at a fundamental level: the LINT-game [2,3].

The rest of the paper is structured as follows: Section 2 shortly introduces the used LINT-game. Section 3 and 4 introduce the evaluation methods and diversity measures used respectively, while section 5 describes how these are combined into testable algorithms. The results are presented in section 6 and section 7 and 8 are used to discuss the results and draw conclusions, in that order.

2 Test Problem: LINT

Watson and Pollack recognized that the mechanics behind the benefits and probelms associated with co-evolution are often still poorly understood, as the evolutionary interaction between individuals and a set of test conditions adds a whole new level of complexity. Therefore they introduced the numbers game as a means to investigate the properties of the co-evolutionary setup in isolation from problem specific difficulties [2]. This class of games as trivially simple as evolving and maximizing integers implemented in a co-evolutionary algorithm displayed archetypal co-evolutionary pathologies, thereby effectively attributing these pathologies to the co-evolutionary setup rather than problem complexity.

For this investigation the Locally Intransitive (LINT) game is used; a specific numbers game introduced by Watson [4] and further investigated by De Jong [3,5]. To avoid complications due to overspecialization the LINT-game used is limited to one dimension and individuals are represented as a single positive floating-point value. A game between a learner L and a test T can be expressed as follows:

$$G(L,T) = \begin{cases} 1 \text{ if } T \geq L \wedge \neg(T \geq (1+\delta)L) \text{ or } \neg(T \geq (1-\delta)L) \\ 0 \text{ otherwise} \end{cases}$$

We can see the LINT-game is locally deceptive as information gained from opponents that fall within the region defined by α gives evolutionary incentives diverging from the global goal. The objective fitness of an individual can now be expressed as: $obj\_fitness(L) = (1 - \delta)L + (1 + \delta)L - L = L$.

To avoid further complications such as disengagement, a single population is used which means that individuals are evaluated on two different roles. On the one hand individuals are evaluated as learners that try to beat as many opponents as possible and on the other hand individuals are evaluated as tests that provide information for the other individuals, see [3].

3 Evaluation Methods

The evaluation of individuals deals with the question how to translate the information gained from interactions between individuals into a meaningful subjective fitness score. In principal sampling randomly selected opponents as a basis for this evaluation is sufficient since the average of the resulting outcomes is guaranteed eventually to converge to the expected outcome. However, this seems like a waste of computational effort, as substantial knowledge is gained during evolutionary runs besides the outcomes of individual interactions. Therefore four evaluation methods are tested that use the information gained from the interactions differently than the most basic interpretation that simply uses the sum of a sequence of interactions as a subjective fitness score.

The **Exponential Moving Average** (EMA) is used in statistics and finance to smooth out short-term fluctuations and study underlying trends [6]. In an evolutionary setup this means that individuals carry not only information about performance in the current timestep, but information about previous timesteps as well, constituting a form of *memory*. Over time the impact of previous timesteps degrades exponentially based on the *smoothing factor* α. Given the actual measured performance Y_t at timestep t and the current EMA score S_t, the new EMA score can be calculated as: $S_{t+1} = \alpha Y_t + (1 - \alpha)S_t$, with α between 0 and 1. For our experiments two different implementations of the EMA method are used: one that updates the EMA score after a cycle of interactions, or tournament, which will be referred to as tEMA and one that updates per individual interaction, which will be referred to as iEMA.

The **Elo-rating**, originally invented by Arpad Elo, is in its current adaptation used as a rating system for chess players [7]. This method aims at rating players in a competitive environment more fairly than counting the total number of wins and losses by attempting to statistically estimate the underlying *true skill* of these players. To do this the current rating of players is taken into account, which means that the impact of a specific match is relative to the players. Given two players A and B with respective ratings R_A and R_B the new rating R'_A of player A can be calculated by means of the *expected* outcome E_A and the *realized* outcome S_A as follows:

- $E_A = \frac{1}{1+10^{(R_B - R_A)/400}}$
- $R'_A = R_A + K(S_A - E_A)$, with K as the maximum rating adjustment

The rating of player B can be updated noting that, $E_A+E_B = 1$ and $S_A+S_B = 1$ New players are initialized with a fixed starting rating.

The **Glicko-rating** is based on the Elo-rating method and originally invented by Mark E. Glickman [8] to provide an alternative rating system for competitive communities. Glickman realized that in the Elo-rating system the impact of a game depends solely on the ratings of both players, the number of previous games played is irrelevant while it seems reasonable that the rating of an individual that has played many games is a more accurate approximation of the underlying true skill. To take the amount of previous games into account Glickman introduced the *Rating Deviation* (RD) that reflects the stability of a players rating and generally decreases as more games are played. Given a player A with rating R_A and rating deviation RD_A we can update based on a game played against opponent B with rating R_B and rating deviation RD_B with outcome S as follows:

- $R' = R + \frac{q}{1/RD_A^2+1/d^2} g(RD_B)(S - E(S|R_A, R_B, RD_B)$
- $RD' = \sqrt{(\frac{1}{RD_A^2} + \frac{1}{d^2})^{-1}}$

With:

- $q = \frac{ln(10)}{400}$
- $g(RD_A) = \frac{1}{\sqrt{1+3q^2(RD_A^2)/\pi^2}}$
- $E(S|R_A, R_B, RD_B) = \frac{1}{1+10^{-g(RD_B)(R_A-R_B)/400}}$
- $d^2 = \frac{1}{q^2 g(RD_B)^2 (E(S|R_A,R_B,RD_B)(1-E(S|R_A,R_B,RD_B)}$

New individuals are initialized with a fixed rating and RD.

4 Diversity Maintenance

A genetically diverse population is more proficient at exploring the fitness landscape, which is generally considered to be beneficial in evolutionary computation. For co-evolutionary algorithms diversity is even more important as the set of test conditions is in itself a dynamic population. In the LINT game specifically, diversity is an important issue as convergence will lead to misleading evolutionary incentives when more and more individuals fall within each others regions of intransitivity. For this reason we also look at two different methods of promoting diversity and see how this influences the performance of the different evaluation methods.

Firstly, the key to a successful test set is to provide learners with relevant challenges so that an informed decision can be made about the relative fitness of individuals. This in general means that a test that makes more *distinctions* between learners carries more information about the desired evolutionary gradient towards the solution concept. A test is considered to make a distinction between two learners if the outcome against one is higher than against the other as in [9]. By using Competitive Fitness Sharing [10] an *informativeness* score can be determined for individuals that expresses the number of distinctions an individual makes weighted by the total number of times every distinction has been made. This score can be used to augment the test-score of an individual and contributes to a more diverse and informative population.

Secondly, the problem of local intransitivity would be solved if all individuals would be spaced apart such that none falls within an others region of intransitivity. Therefore the challenge posed by the LINT game can be viewed as a dynamic multi-modal search problem [11], as the challenge is not only to find the individual with the highest numerical value, but also to maintain lower valued individuals along the way, spaced to avoid overlap and thereby sustain progress. In a sense a non-overlapping individuals become desirable local optima. Various methods have been proposed to deal with multi-modal environments, but for this paper elements of *Deterministic Crowding* (DC) [12] and *Restricted Tournament Selection* (RTS) [13] are used. New individuals enter competition for space with one of the existing individuals based on *phenotypic similarity* and new individual can replace the old one if it has a higher subjective fitness score. To measure phenotypic similarity the outcomes of games played by an individual in the current evolutionary cycle are recorded in an *outcomevector*, and the Hamming distance between two outcomevectors is used to express similarity.

5 Algorithmic Details

Using the above descriptions of diversity maintenance three algorithms can be constructed to test the five evaluation methods introduced in section three; the **Basic** algorithm with no specific diversity operator, the **Informative** algorithm and the **Multi-Modal** algorithm. The specific implementation is based on De Jong [3] and is depicted in Fig. 1 and operates as follows:

```
1. $pop := initialize\_random();$
2. **for** $gen := 1 : generations\{$
3.     $pop := pop \cup mutate(pop);$
4.     **for** $i := 1 : |pop|\{$
5.         $learner\_score_i := \sum_j^{|pop|}(G(pop_i, pop_j);$
6.         $test\_score_i := \sum_j^{|pop|}(1 - G(pop_j, pop_i));$
7.     $\}$
8.     $sub\_fitness_i := learner\_score_i + test\_score_i;$
9.     $pop := select(pop, sub\_fitness);$
10.$\}$
```

Fig. 1. Pseudo-code for the Basic algorithm using the base evaluation method

initialize_random(). An initial population of size n is generated as a set of uniformly distributed floating-point values between 0 and 1. Necessary other scalars are initialized to the appropriate value; the base and EMA evaluation methods start with a subjective fitness score of 0, while the Elo- and Glicko-rating methods start with a rating of 1500 (both) and a RD of 350 (for Glicko only).

mutate. In this step n new individuals are generated by randomly selecting individuals from the current population without replacement and adjusting the floating-point value by a normally distributed random value with mean -0.025 and standard deviation 0.1. The negative mutational bias reflects the idea that a random mutation is generally more harmful than beneficial [3].

evaluate. Every individual in the new population of size $2n$ is played against all other individuals twice: once as a learner and once as a test. Based on this cycle of interactions learner- and test-scores are calculated and combined into a subjective fitness score. For the Basic algorithm the tEMA evaluation method calculates the subjective fitness differently by replacing line 8 in the pseudo-code by:

8. $sub\_fitness_{i\ new} := \alpha(learner\_score_i + test\_score_i) + (1-\alpha)sub\_fitness_{i\ old}$;

For iEMA every individual interaction is used to update both the learner- and test-score and the Elo and Glicko methods use player ratings as the measure of subjective fitness and update accordingly.

The Informative algorithm uses the informativeness of individuals to augment the test-score of these individuals. The parameter γ determines how much influence the informativeness has on the final score. In this case line 6 should read:

6. $test\_score_i := \gamma(\sum_j^{|pop|}(1 - G(pop_j, pop_i))) + (1-\gamma) * inf_i$;

The informativeness score is based on a sequence of interactions and therefore poses a problem for evaluation methods that update per interaction. For iEMA this is solved by normalizing the informativeness score and adding this to the test-score appropriately:

$test\_score_i = \gamma * test\_score_i + (1-\gamma) * inf_i/inf_{\max}$

For the Elo-rating method the sum of the expected and realized outcomes over this sequence of interactions is therefore used as follows:

$R'_i = R_i + K((\gamma \sum S_i + (1-\gamma) * inf_i) - \sum E_i)$

Since it is very hard to integrate the informativeness into the Glicko updating process in a meaningful way, this has not been implemented for the Informative algorithm. For the Multi-Modal algorithm all subjective fitness scores are calculated as for the Basic algorithm and scorevectors of the individual players are recorded, representing phenotypic behavior.

select. For both Basic and Informative algorithms ranked selection with replacement is used to construct a new population of n individuals, where the sorting rank is used as the relative selection probability. For the Multi-Modal algorithm every mutant may try to replace one original individual that is closest phenotypically. The original individual is replaced if the subjective fitness score of the mutant is higher. If more than one mutant compete with a single individual the best mutant is chosen.

The Basic, Informative and Multi-Modal algorithms are run for 500, 1500 and 5000 generations respectively in order to study the qualitative behavior over time. All results presented in the next section are averaged over 100 runs and differences in performance are considered significant if $p < 0.001$ using the Mann-Whitney U-test. In all cases the population size is 20 and δ is 0.05. As observables, the average objective fitness of a population and the OFC are used, where the OFC is calculated as the Pearson correlation coefficient between the objective and subjective fitness values of individuals within a population [3].

6 Experimental Results

The results for the **Basic** algorithm with respect to the average objective fitness are presented in Fig. 2a. The parameters for the Elo method (K) and both EMA methods (α) are set to 32 and 0.9 respectively. As we can clearly see all evaluation methods increase in performance rapidly at first, but slowly peter out as time progresses. Although both Elo and iEMA significantly outperform the base method and both Glicko and tEMA perform significantly worse, the differences are small and the behavior of the methods is very similar; stable progress cannot be maintained over time. This observation is confirmed by the measure OFC's displayed in Fig. 2b, as the OFC's degrade to values fluctuating around 0 within the same time-frame as the stagnation of performance.

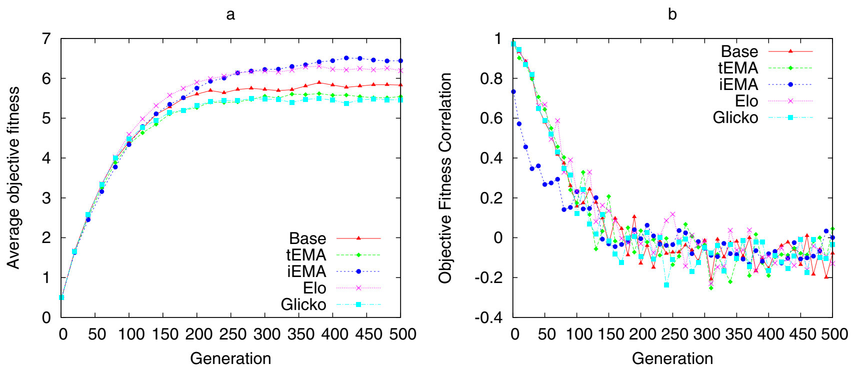


Fig. 2. a.) Average objective fitness and b.) OFC over time for the Basic algorithm

The results for the **Informative** algorithm are presented in Fig. 6. The parameters K and α are again set to 32 and 0.9 respectively, while γ is set to 0.6 for the base and Elo evaluation methods and set to 0.5 for both EMA methods. The iEMA method is outperformed significantly by all other methods and actually performs similar to iEMA in combination with the Basic algorithm. Apparently iEMA does not benefit from the added informativeness score, which makes sense as iEMA essentially represents a form of random sampling and unlikely to co-operate well with the global measure of informativeness. On the other hand the other three methods perform much better than for the Basic algorithm, with tEMA outperforming Elo significantly. However, again the results are close together and the gross behavior is comparable to the Basic algorithm, being unable to sustain progress. This process is closely mirrored by the measured OFC's.

Figure 6 displays the results for the **Multi-Modal** algorithm. The parameters k and α are set to 16 and 0.1 respectively. Opposed to the previous algorithms the differences between the evaluation methods are large, in fact all differences are highly significant. The tEMA, Elo and Glicko evaluation methods score much higher than for the other algorithms, especially tEMA. The base evaluation method on the

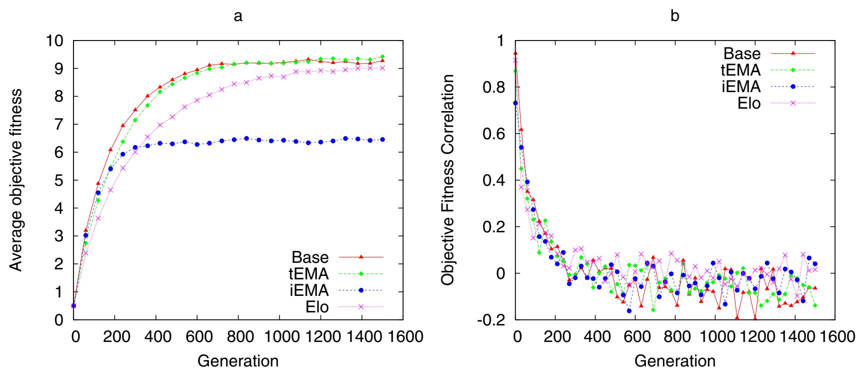


Fig. 3. a.) Average objective fitness and b.) OFC over time for the Informative algorithm

other hand lies between the Informative and Basic algorithm with respect to performance and iEMA performs slightly worse than on both other algorithms. Most notably however, is that for all evaluation methods stable progress is maintained and there is no indication yet that it will not be maintained later on. This is again reflected by the measured OFC's as in Fig. 6b; all OFC-scores remain very high over time, indicating a meaningful correlation between the objective fitnesses and awarded subjective fitnesses within a population.

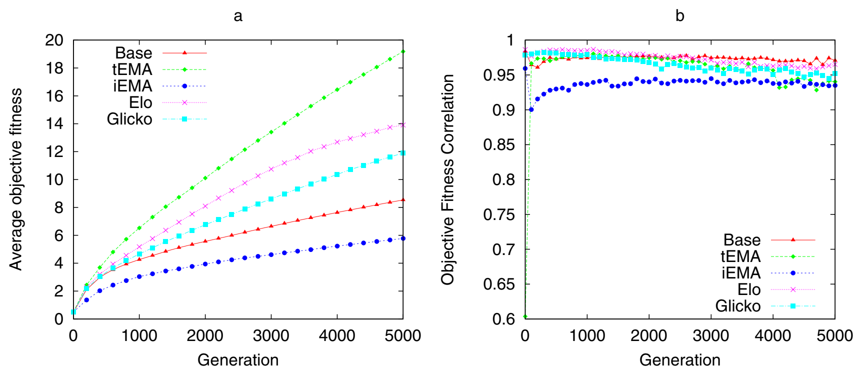


Fig. 4. a.) Average objective fitness and b.) OFC over time for the Multi-Modal algorithm

7 Discussion

The results of both the Basic and the Informative algorithms can be understood in terms of co-evolutionary failure; both systems eventually reach an equilibrium state where the averaged performance remains at a constant level. The introduction of the informativeness as an extra measure of performance as a test definitively improves the results for evaluation methods to which it can be appropriately applied, but in the end the problem of local intransitivity still prevents stable progress. In the single population set-up there is a trade-off between the informativeness score and the regular outcomes as the first leads to a more diverse and informative set and the latter provides directed evolutionary pressure. Such a trade-off is not explicitly present in the Multi-Modal algorithm, as the alternative method of selection operates separately from the calculation of fitness. While the algorithm progresses slowly compared to the other two algorithms, as the replacement of individuals is done in a more conservative way, progress is maintained over time. This means that the population is kept at a level of diversity that is meaningful and enables adequate evaluation. By using the outcomevector as a measure of similarity behavioral *niches* are protected and the implicit connection between the genotypically determined objective fitness and the outcomevector of individuals guarantees genotypic diversity to a certain degree.

More generally we can see that for this problem the performance of a specific evaluation method cannot be viewed in isolation, it strongly depends on the specific algorithmic implementation, in this case the degree of diversity maintenance. Also we found no evaluation method that is in itself capable of overcoming the obstacle of local intransitivity and in fact in the case of co-evolutionary failure as in the Basic and Informative algorithms the final result associated with an evaluation method is dictated more by the failure of the algorithm than by the method itself, as these final results lie very close. If co-evolutionary failure is staved off however, differences in the evaluation methods do become apparent and in the case of the Multi-Modal algorithm the methods that take outcomes from previous generations into account (tEMA, Elo and Glicko) perform much better. The more complex Elo and Glicko evaluation methods perform worse than tEMA, so the idea that these more statistically inspired methods could lead to better estimations of the objective fitness is not empirically validated for this problem.

8 Conclusion

Although the LINT-game was used to eliminate as much complicating factors as possible, performance of the various evaluation methods is still dependent on the specific diversity method. We found no single evaluation method that performs qualitatively different than the rest under the same algorithmic assumptions. What we did find is that the Multi-Modal approach to the LINT-problem proved very effective in terms of stable performance although not as efficient for all evaluation methods. Lastly, the OFC has been shown to be an adequate tool for monitoring co-evolutionary dynamics with respect to accuracy of evaluation.

More specifically the idea of approaching a search problem with multi-modal techniques to preserve functional diversity should be explored more thoroughly, using different test problems and a broader range of parameters. This would lead to a better understanding of the dynamics governing the formation and preservation of critical niches.

References

1. Sevan, G.: Ficici. Solution Concepts in Coevolutionary Algorithms. PhD thesis, Brandeis University (2004)
2. Watson, R.A., Pollack, J.B.: Coevolutionary dynamics in an minimal substrate. In: Spector, L., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001, pp. 702–709. Morgan Kaufman, San Fransisco (2001)
3. De Jong, E.D.: Objective fitness correlation. In: Thierens, D., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2007, pp. 440–447. ACM press, New York (2007)
4. Watson, R.A.: Personal communication (2003)
5. de Jong, E.D.: Intransitivity in Coevolution. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 843–851. Springer, Heidelberg (2004)
6. Chou, Y.-l.: Statistical Analysis. Holt International (1975) ISBN 0030894220
7. Elo, A.E.: The Rating of Chessplayers: Past and Present. Arco (1978)
8. Glickman, M.E.: Parameter estimation in large dynamic paired comparison experiments. Applied Statistics 48(3), 377–394 (1999)
9. Ficici, S.G., Pollack, J.B.: Pareto optimality in coevolutionary learning. In: Kelemen, J., Sosík, P. (eds.) ECAL 2001. LNCS (LNAI), vol. 2159, pp. 316–325. Springer, Heidelberg (2001)
10. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. Evolutionary Computation 5(1), 1–29 (1997)
11. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing. Springer, Berlin (2003)
12. Mahfoud, S.W.: Crowding and preselection revisited. In: Manner, R., Manderick, B. (eds.) Parallel Problem Solving from Nature 2, PPSN-II (2nd PPSN 1992), pp. 27–36. Elsevier Science Publishers, Amsterdam (1992)
13. Harik, G.R.: Finding multimodal solutions using restricted tournament selection. In: Eshelman, L.J. (ed.) Proc. of the Sixth Int. Conf. on Genetic Algorithms, pp. 24–31. Morgan Kaufmann, San Francisco (1995)

Comparison of Adaptive Approaches for Differential Evolution

Karin Zielinski, Xinwei Wang, and Rainer Laur

Institute for Electromagnetic Theory and Microelectronics (ITEM),
University of Bremen, P.O. Box 330440, 28334 Bremen, Germany
zielinski@item.uni-bremen.de

Abstract. The evaluation of optimization algorithms and especially the analysis of adaptive variants is often complicated because several features are modified concurrently. For Differential Evolution these features may be adaptation of parameters, adjustment of the strategy and addition of local search or other special operators. Thus, it is difficult to analyze which of these procedures is actually responsible for changes in the performance. Therefore, in this work several adaptive algorithms are studied in-depth by monitoring performance changes for individual components of these algorithms to examine their effectiveness. The results show among others that the performance can be significantly improved by employing strategy control.

1 Introduction

Differential Evolution (DE) [1] is a relatively new evolutionary algorithm that is often used in literature because of its fast convergence behavior. Another property that is often associated with DE is ease of use as it only contains three control parameters: The population size *NP*, the scaling factor F that is used in mutation, and the crossover probability *CR*. For F and *CR* the range of recommended values given in literature is mostly relatively small. However, recommended settings for *CR* depend on the decomposability of the objective function which is a property that is not necessarily known in advance, and furthermore for multi-objective optimization suitable settings may be different from single-objective optimization. Another decision associated with applying DE is which strategy to use. Often this question is not discussed in the literature, and in most papers DE/rand/1/bin is used [2] which actually seems to be a good choice for many optimization problems, but there are also examinations in the literature which conclude that other strategies perform better [2,3].

In order to overcome these problems with having to choose control parameter settings and a suitable DE strategy, several adaptive variants of DE have been presented in the literature where one or several of the control parameters and/or the DE strategy are modified during an optimization run [4,5,6,7,8,9,10,11,12,13]. However, generally several features are modified concurrently, thus it is not clear which of the features are actually effective and which ones only complicate the algorithm without contributing anything to its performance. Therefore, a detailed analysis of several adaptive approaches from literature is done here by regarding individual components of these algorithms separately, based on an extensive test set of constrained single-objective optimization problems.

This paper is organized as follows: In Section 2 the Differential Evolution algorithm, its control parameters and several strategies for DE are introduced. The adaptive approaches which will be examined in this work are described in Section 3. Experimental settings are given in Section 4, results are discussed in Section 5, and the paper ends with conclusions in Section 6.

2 Differential Evolution

Differential Evolution can be classified as a population-based stochastic evolutionary optimization algorithm. As in other evolutionary algorithms the first generation is initialized randomly in a given search space, and further generations evolve by applying specific evolutionary operators until a stopping criterion is satisfied. The DE individuals consist of real-valued vectors with dimension D that equals the number of objective function parameters. The number of individuals NP has to be set by the user. In every generation one offspring is generated for each population member $\boldsymbol{x}_i$ (with $i \in \{0, \dots, NP-1\}$) using the evolutionary operators mutation and recombination.

There are several strategies for DE that differ in the way mutation and recombination is conducted [1,2]. They have the commonality that during mutation a mutated vector $\boldsymbol{v}_i$ is generated that is a linear combination of several individuals, and during recombination components from the mutated vector and the target vector $\boldsymbol{x}_i$ (which is a population member) are combined to build the trial vector $\boldsymbol{u}_i$ that competes with $\boldsymbol{x}_i$ for a place in the next generation during selection.

The variants are specified using the notation DE/x/y/z where x denotes the vector to be mutated, y is the number of difference vectors and z is the crossover scheme [14]. The vector to be mutated (also called base vector) might be a randomly chosen vector (notation: 'rand'), the best vector that was found so far (notation: 'best'), or a vector that is located on the connecting line between two solutions, e.g. between the target vector and a random vector (notation: 'current-to-rand') or between the target vector and the best vector (notation: 'current-to-best'). The number of difference vectors y is normally set to one or two. Concerning the crossover scheme, a binomial or exponential process can be used (notation: 'bin' or 'exp', respectively) which differ in the way that components are chosen for copying to the trial vector [14]. In [14] the use of binomial crossover is recommended but in [15] it is stated that there are no significant differences between the crossover methods, so in this work only the binomial process is used that will be explained later in this section.

The strategies which will be used in this work are given in Table 1 where F and K are control parameters of DE ($K = F$ is always assumed in this work as it is also often done in the literature, e.g. in [16]), the number of difference vectors is set to $y = \{1,2\}$, the indices $r_1, \dots, r_5$ denote mutually different individuals which are also different from $\boldsymbol{x}_i$, and $\boldsymbol{x}^*$ is the best individual found so far.

Table 1. Strategies of Differential Evolution

Notation	Equation for mutated vector
DE/rand/y	$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F \cdot \sum_{m=0}^{y-1} \left(\boldsymbol{x}_{r_{2+2\cdot m}} - \boldsymbol{x}_{r_{3+2\cdot m}}\right)$
DE/current-to-rand/y	$\boldsymbol{v}_i = \boldsymbol{x}_i + K \cdot (\boldsymbol{x}_{r_1} - \boldsymbol{x}_i) + F \cdot \sum_{m=0}^{y-1} \left(\boldsymbol{x}_{r_{2+2\cdot m}} - \boldsymbol{x}_{r_{3+2\cdot m}}\right)$
DE/best/y	$\boldsymbol{v}_i = \boldsymbol{x}^* + F \cdot \sum_{m=0}^{y-1} \left(\boldsymbol{x}_{r_{1+2\cdot m}} - \boldsymbol{x}_{r_{2+2\cdot m}}\right)$
DE/current-to-best/y	$\boldsymbol{v}_i = \boldsymbol{x}_i + K \cdot (\boldsymbol{x}^* - \boldsymbol{x}_i) + F \cdot \sum_{m=0}^{y-1} \left(\boldsymbol{x}_{r_{1+2\cdot m}} - \boldsymbol{x}_{r_{2+2\cdot m}}\right)$

In this work binomial recombination is used for every strategy, thus trial vectors $\boldsymbol{u}_i$ are built (with $i \in \{0, \ldots, NP-1\}$) by determining for every vector component $j \in \{0, \ldots, D-1\}$ if the corresponding component should be copied from the target vector $\boldsymbol{x}_i$ or the mutated vector $\boldsymbol{v}_i$. The decision is made using a random variable $rand_j$ that is compared with the control parameter CR. However, because during selection $\boldsymbol{u}_i$ and $\boldsymbol{x}_i$ will be compared, it is ensured for every individual that at least one component of $\boldsymbol{u}_i$ is derived from $\boldsymbol{v}_i$ by a random choice of a number $k \in \{0, \ldots, D-1\}$:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } rand_j \leq CR \text{ or } j = k \\ x_{i,j} & \text{otherwise} \end{cases} \tag{1}$$

The selection process is identical for all strategies. Selection is conducted by comparing the target vector $\boldsymbol{x}_i$ with the trial vector $\boldsymbol{u}_i$. For unconstrained single-objective minimization problems, the solution that yields the smaller objective function value is chosen for the next generation ($\boldsymbol{u}_i$ might also be preferred in case of equality to enable crossing of flat regions in objective space). However, for constrained optimization problems the selection procedure has to be modified. In this work the feasibility rules described in [17] are applied, thus when a solution $\boldsymbol{a}$ is compared to a solution $\boldsymbol{b}$, $\boldsymbol{a}$ is considered better if:

– Both solutions are feasible, but $\boldsymbol{a}$ yields the smaller objective function value.
– $\boldsymbol{a}$ is feasible and $\boldsymbol{b}$ is not.
– Both solutions are infeasible, but $\boldsymbol{a}$ has the lower sum of constraint violations.

For easier comparability this technique is used throughout this work although some of the algorithms examined here originally use slightly different methods for constraint-handling. The same holds for boundary constraints: To simplify this analysis, a limit-exceeding parameter is set to the middle between the old position and the boundary for all algorithms, regardless of the original approach (other methods may be examined in future work).

3 Adaptive Approaches

There are several adaptive approaches in the DE literature. In this paper three of them are considered in detail: jDE [4], SaDE [5] and DE_DoE [6]. Other methods like described in [7,8,9,10,11,12,13] may be regarded in future work.

3.1 jDE

A self-adaptive DE algorithm called jDE is described in [4], and its extension jDE-2 is presented in [18,19]. Each individual $\boldsymbol{x}_{i,G}$ has its own values of control parameters, denoted $F_{i,G}$ and $CR_{i,G}$ (where G is the current generation). The control parameters are updated as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u & \text{if } rand_2 < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases} \tag{2}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \tag{3}$$

where $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$, $F_u = 0.9$, and the random numbers $rand_j \in [0, 1]$ (with $j \in \{1, 2, 3, 4\}$) are from a uniform distribution. Thus, $F \in [0.1, 1.0]$ and $CR \in [0, 1]$ are adapted based on the probabilities τ_1 and τ_2. The jDE and jDE-2 algorithm differ in the handling of boundary constraints, but to limit the complexity of this examination, only one method for handling boundary constraints is used here (see Section 2). Furthermore, in contrast to the jDE algorithm that uses only strategy DE/rand/1/bin [4], in jDE-2 three strategies are used (DE/rand/1/bin, DE/current-to-best/1/bin, DE/rand/2/bin), and each individual has separate control parameters for each strategy. A further modification of the jDE-2 algorithm is that the k worst individuals are replaced every l generations with randomly chosen positions where $l = 1000$ and $k = 70$ in [18]. For constraint-handling, the feasibility rules by Deb are used in [18,19] whereas in [4] unconstrained optimization problems are regarded. To enable an easier comparison, this work is limited to considering jDE.

3.2 SaDE

In [5] not only parameter settings but also the DE strategy is changed during an optimization run. Four different strategies are used (DE/rand/1, DE/current-to-best/2, DE/rand/2, DE/current-to-rand/1; in a previous paper only the first two strategies were used [20]) which all have the same probability to be selected in the beginning of an optimization run. Later, the probabilities are modified by calculating for every strategy the ratio r of the number of trial vectors which successfully entered the next generation divided by the number of generated trial vectors. This number is sampled for 20 generations (learning period). Settings for F are randomly taken from a normal distribution with a constant mean of 0.5 and a constant standard deviation of 0.3 (where F is limited to $F \in (0, 2]$). Thus, F is varying but there is no feedback from the search. Because the authors assume that the setting of CR is more sensitive to the optimization problem (in a previous paper they also stated that F is more related to the convergence speed [20]), CR is adapted based on feedback from the search. Like F, it is also selected from a normal distribution with a fixed standard deviation (0.1), but the mean CRm is adapted. In the beginning $CRm = 0.5$ is used, and each individual is randomly assigned a CR value that is kept constant for five generations and then a new value is randomly chosen using the same distribution. After 20 generations, CRm is recalculated based on r, and the described procedure is repeated. To speed up convergence, in [5] a local search method (Sequential Quadratic Programming) is used every 500 generations. To limit the complexity of the present examination, the local search is omitted here. Constraints are handled slightly differently in [5] than presented here, but to simplify the analysis the constraint-handling method as described in Section 2 is employed throughout this work.

3.3 DE_DoE

In [6] the adaptation of F and CR based on methods from Design of Experiments (DoE) [21] is shown for constrained single-objective optimization, and in [22] the method is extended for multi-objective optimization. Significant differences in performance of different parameter settings can be detected by analysis of variance (ANOVA) which is associated with a confidence coefficient α that is set to $\alpha = 0.01$ here. Performance is measured similarly to SaDE (for single-objective optimization r is also multiplied with the improvement of the objective function value). Due to sophisticated designs, not only main effects but also interaction effects can be detected while keeping the computational cost low. In this work a two-level factorial design is employed, therefore two settings of F and CR are regarded at any given time, respectively, and initial settings of $F = \{0.7, 0.9\}$ and $CR = \{0.2, 0.9\}$ are used. Each parameter combination is applied to one fourth of the individuals in every generation (therefore, NP must be divisible by four). Each generation equals a replicate of the DoE analysis, and if a significant effect has been found, the parameters are adapted (in this case the worse performing setting is changed by 0.1 in the direction of the better one, and the better performing setting changes by 0.05 in the same direction), and the DoE analysis is restarted. The analysis is also restarted if no significant effects occurred for some time (in this case for 10 generations) because the performance may vary over time, complicating the detection of significant effects.

4 Experimental Settings

In order to examine which components of the adaptive approaches are especially effective, the following algorithm variants are considered in this work:

- V1: DE_DoE
- V2: jDE'
- V3: SaDE'
- V4: jDE' with strategy control as in SaDE'
- V5: SaDE' with adaptive F
- V6: SaDE' without strategy control
- V7: SaDE' with eight strategies
- V8: DE_DoE with strategy control as in SaDE'

Here, V1 is the DE_DoE algorithm used in [6], whereas V2 and V3 are close to the original jDE and SaDE algorithms given in [4] and [5], respectively (they are not equal because the same constraint-handling and the same handling for boundary constraints is used for all algorithm variants here as described in Section 2, and local search is omitted). In V4 the jDE' algorithm is used for parameter control while a strategy control as in SaDE' is also used. In V5 SaDE' is used as basis and F is adaptively controlled in the same way as CR (with an initial mean of $Fm = 0.5$ and a standard deviation of 0.3, like suggested in [5]). To examine the influence of strategy control on SaDE', V6 only uses the parameter control of SaDE' without strategy control. Because several more strategies exist besides the ones employed in SaDE', in V7 all strategies given in Table 1 are used (with $y = \{1, 2\}$). In V8 the strategy control of SaDE' is employed in DE_DoE.

The same population size is used for all algorithm variants. Because $NP = 50$ was used in [5], $NP = 50$ for $D = 10$ and $NP = 100$ for $D = 30$ in [20], $NP = 100$ in [4], $NP = 200$ in [18,19] and $NP = 52$ in [6] (because NP must be divisible by four for this method), in this work $NP = 52$ is used. The maximum number of function evaluations is 500,000.

In [23] 24 constrained single-objective test problems with a large variety of different features have been defined for the Special Session on Constrained Real Parameter Optimization at the Congress on Evolutionary Computation 2006. This test set is used as basis for the examination (where functions g20, g22 and g23 have been omitted because none of the here examined algorithms was able to find the optimal solution). Thus, 21 functions are employed, for which 25 optimization runs have been done for every algorithm variant, respectively.

The following performance measures are used: The success rate sr is the percentage of runs in which the global optimum has been found (with an accuracy of 10^{-4}). Because the success performance sp as given in [23] (average number of function evaluations for convergence divided by sr) may lead to wrong conclusions when sr is low, the average number of function evaluations for convergence is regarded here. The feasible rate fr (percentage of runs in which at least one feasible individual was found) is omitted because almost every optimization run resulted in feasible individuals (exceptions are V1 with $fr = 96\%$ for g13 and $fr = 92\%$ for g17, V3 with $fr = 44\%$ for g13, and V8 with $fr = 92\%$ for g14).

5 Results

The success rate of all algorithm variants is given in Table 2, and the average number of function evaluations for convergence is shown in Table 3[1].

To evaluate the different methods of parameter control, the strategy control of SaDE' has been disabled for V6, and results for V6 are compared with jDE' (V2) and DE_DoE (V1). The success rate of V6 is better than for V1 and V2 for many functions although there are also exceptions (g02, g14, g19). The average number of function evaluations for convergence is also often comparable to or better than the results for V1 and V2. It is concluded that the parameter control of SaDE' is generally superior to the methods employed in jDE' and DE_DoE.

[1] Due to space limitations only partial results can be shown here. More detailed results are available at http://www.item.uni-bremen.de/staff/zilli/PPSN08_results.xls

Table 2. Success rate in %

Problem	V1	V2	V3	V4	V5	V6	V7	V8
g01	100	100	100	100	100	100	100	100
g02	76	72	44	88	20	28	16	84
g03	0	0	16	0	92	100	92	0
g04	100	100	100	100	100	100	100	100
g05	44	32	100	88	100	92	64	84
g06	100	100	100	100	100	100	100	100
g07	60	96	100	100	100	100	4	72
g08	100	100	100	100	100	100	100	100
g09	96	100	100	100	100	100	100	100
g10	28	40	100	96	100	100	0	76
g11	92	100	100	100	100	100	100	100
g12	100	100	100	100	100	100	100	100
g13	0	0	16	0	44	4	24	0
g14	52	0	100	100	44	4	0	72
g15	92	60	100	100	100	100	48	92
g16	100	100	100	100	100	100	100	100
g17	4	0	36	4	32	4	8	8
g18	68	84	84	80	72	72	60	80
g19	28	24	48	100	60	4	0	72
g21	20	60	68	80	92	68	36	40
g24	100	100	100	100	100	100	100	100

Furthermore, it is tested if advantages can be gained by controlling F adaptively in the same way as done for *CR* (V5), therefore variants V3 (SaDE') and V5 are compared. Regarding the success rate, it depends on the function if a better performance is reached. For g03, g13, g19 and g21 the performance is better with adaptive F whereas for g02, g14 and g18 the performance becomes worse. The average number of function evaluations for convergence is better for g02, g03, g09, g16 and g18 with adaptive F, and it is worse for g05, g07, g10, g13, g14, g15, g17, g19 and g21 (functions for which the results are similar are not specified here). Therefore, although the success rate averaged over all functions can be slightly improved by using an adaptive F (from 78% to 80%), the overall performance tends to become worse when considering the average number of function evaluations for convergence.

To analyze the influence of strategy control, a comparison of each algorithm with and without strategy control is done. For the jDE' algorithm (V2 without strategy control and V4 with strategy control) the strategy control clearly results in improved performance concerning the success rate as it is always equal or higher for V4 (with exception of g18 where the success rate is slightly better without strategy control). The average number of function evaluations for convergence is often in a similar range. For some functions V2 is better (g06, g11, g18, g21, g24) and for other functions V4 is better (g02, g07, g09, g10, g15, g19), so no definite conclusion can be given regarding this performance measure.

Table 3. Average number of function evaluations for convergence

Problem	V1	V2	V3	V4	V5	V6	V7	V8
g01	28180	20605	24277	22902	24005	20221	18039	25165
g02	194869	68767	447694	62372	37901	206936	325195	182247
g03	-	-	327083	-	280131	151734	151789	-
g04	37488	27761	15868	29775	14792	16782	11345	17789
g05	255793	228783	102745	239549	171724	250927	176413	247343
g06	16858	13521	9416	16348	9238	8530	6827	12061
g07	199319	210108	51793	128084	97990	159242	64946	172512
g08	1356	1277	1491	1415	1445	1286	816	1338
g09	127950	49097	16077	43141	14200	19637	11662	46958
g10	233591	325477	60265	224051	164689	244380	-	171405
g11	41082	29114	22542	45232	21391	24357	10154	63000
g12	2313	2041	2002	2333	2044	1822	1448	2182
g13	-	-	247711	-	316994	373682	281284	-
g14	284388	-	47177	163763	321290	439743	-	126777
g15	173911	270119	60450	162129	98147	164098	280830	216409
g16	18812	15368	24880	16064	10777	25068	18222	15286
g17	406848	-	255101	431151	331256	443380	154002	275375
g18	111613	87950	40206	115294	31143	43680	17171	100533
g19	318225	351489	83244	246529	218128	396186	-	241520
g21	266365	151667	108141	174406	139919	171408	131463	185859
g24	4484	4823	4119	5321	3814	3767	3260	4124

Comparing SaDE' with and without strategy control (V3 and V6, respectively), the success rate with strategy control is better than or equal to the variant without strategy control except for function g03. Concerning the average number of function evaluations for convergence, V3 is also mostly better than V6 (for g05, g07, g09, g10, g13, g14, g15, g17, g19 and g21 whereas V3 was worse for g02 and g03).

For DE_DoE the variant V1 is compared with V8 that uses the strategy control of SaDE'. The success rate of V8 is always better than or equal to the results of V1. Concerning the average number of function evaluations for convergence, also an improvement can be seen for nearly every function when using strategy control (exceptions are only functions g11 and g15).

It can be concluded that strategy control is able to improve the results for all regarded algorithms, especially concerning the success rate. To analyze the effect of using more than the four strategies given in [5], variant V3 (SaDE') is compared to variant V7 that uses the parameter and strategy control described in SaDE' but with eight instead of four strategies. However, the results using V7 deteriorate for many functions concerning the success rate (exceptions are g03 and g13). Interestingly, the average number of function evaluations for convergence improves for several functions using eight instead of four strategies (exceptions are only g05, g07, g13, g15 and g21). The reason for this behavior is not yet clear. For future work the probabilities of the strategies should be monitored during optimization runs to develop a theory for this behavior.

6 Conclusions

The comparison of jDE', SaDE' without strategy control and DE_DoE showed that the parameter control employed in SaDE' is the most efficient strategy which may be explained by the use of different control parameters for each individual and the adjustment of *CRm* based on feedback from the search. Adapting not only *CR* but also F does not have a large effect as assumed in [5]. Nevertheless, it would be interesting to observe the development of *CRm* and Fm over time in future work.

Strategy control had a positive effect on the success rate of jDE', SaDE' and DE_DoE. The average number of function evaluations for convergence also mostly improved but concerning this performance measure the results were less clear (except for DE_DoE for which the results improved for all but two functions). However, it does not seem to be wise to use too many strategies in strategy control as the results concerning success rate for SaDE' mostly deteriorated when using eight strategies. However, the average number of function evaluations for convergence mostly improved, therefore future work observing the probabilities of the strategies over time should provide more insight into the DE dynamics that cause this behavior.

In future work also the local search procedure employed in SaDE and the random substitution used in jDE-2 should be analyzed. Besides, in Section 3 many other adaptive approaches are mentioned which may also be examined.

References

1. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
2. Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.A.: A Comparative Study of Differential Evolution Variants for Global Optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, Washington, USA, pp. 485–492 (2006)
3. Gämperle, R., Müller, S.D., Koumoutsakos, P.: A Parameter Study for Differential Evolution. In: Grmela, A., Mastorakis, N. (eds.) Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, pp. 293–298. WSEAS Press (2002)
4. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (2006)
5. Huang, V., Qin, A., Suganthan, P.: Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 324–331 (2006)
6. Zielinski, K., Laur, R.: Parameter Adaptation for Differential Evolution with Design of Experiments. In: Proceedings of the IASTED International Conference on Computational Intelligence, San Francisco, USA, pp. 212–217 (2006)
7. Zaharie, D.: Parameter Adaptation in Diffential Evolution by Controlling the Population Diversity. In: Proceedings of the 4th Workshop on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, pp. 385–397 (2002)

8. Abbass, H.A.: The Self-Adaptive Pareto Differential Evolution Algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, pp. 831–836 (2002)
9. Ali, M.M., Törn, A.: Population Set Based Global Optimization Algorithms: Some Modifications and Numerical Studies. Computers and Operations Research 31(10), 1703–1725 (2004)
10. Liu, J., Lampinen, J.: A Fuzzy Adaptive Differential Evolution Algorithm. Soft Computing - A Fusion of Foundations, Methodologies and Applications 9(6), 448–462 (2005)
11. Xue, F., Sanderson, A.C., Bonissone, P.P., Graves, R.J.: Fuzzy Logic Controlled Multi-objective Differential Evolution. In: Proceedings of the IEEE International Conference on Fuzzy Systems, Reno, NV, USA, pp. 720–725 (2005)
12. Teo, J.: Exploring Dynamic Self-adaptive Populations in Differential Evolution. Soft Computing 10(8), 673–686 (2006)
13. Salman, A., Engelbrecht, A.P., Omran, M.G.: Empirical Analysis of Self-Adaptive Differential Evolution. European Journal of Operational Research 183(2), 785–804 (2007)
14. Price, K.V.: An Introduction to Differential Evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 79–108. McGraw-Hill, London (1999)
15. Price, K., Storn, R.: Website as in March 2008, `http://www.icsi.berkeley.edu/~storn/code.html`
16. Storn, R.: Designing Digital Filters with Differential Evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 109–125. McGraw-Hill, London (1999)
17. Deb, K.: An Efficient Constraint Handling Method for Genetic Algorithms. Computer Methods in Applied Mechanics and Engineering 186(2-4), 311–338 (2000)
18. Brest, J., Žumer, V., Maučec, M.S.: Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 919–926 (2006)
19. Brest, J., Žumer, V., Maučec, M.S.: Control Parameters in Self-Adaptive Differential Evolution. In: Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications, Ljubljana, Slovenia, pp. 35–44 (2006)
20. Qin, A., Suganthan, P.: Self-adaptive Differential Evolution Algorithm for Numerical Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, pp. 1785–1791 (2005)
21. Montgomery, D.C.: Design and Analysis of Experiments. John Wiley and Sons, Chichester (2001)
22. Zielinski, K., Laur, R.: Differential Evolution with Adaptive Parameter Setting for Multi-Objective Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, pp. 3585–3592 (2007)
23. Liang, J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello Coello, C.A., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore (March 2006)

Analyzing Hypervolume Indicator Based Algorithms

Dimo Brockhoff[1], Tobias Friedrich[2], and Frank Neumann[2]

[1] Computer Engineering and Networks Lab, ETH Zurich, 8092 Zurich, Switzerland
dimo.brockhoff@tik.ee.ethz.ch
[2] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
firstname.lastname@mpi-inf.mpg.de

Abstract. Indicator-based methods to tackle multiobjective problems have become popular recently, mainly because they allow to incorporate user preferences into the search explicitly. Multiobjective Evolutionary Algorithms (MOEAs) using the hypervolume indicator in particular showed better performance than classical MOEAs in experimental comparisons. In this paper, the use of indicator-based MOEAs is investigated for the first time from a theoretical point of view. We carry out running time analyses for an evolutionary algorithm with a $(\mu + 1)$-selection scheme based on the hypervolume indicator as it is used in most of the recently proposed MOEAs. Our analyses point out two important aspects of the search process. First, we examine how such algorithms can approach the Pareto front. Later on, we point out how they can achieve a good approximation for an exponentially large Pareto front.

1 Introduction

In the last decades, there has been a growing interest in developing evolutionary algorithms for multiobjective optimization problems. Many variants proposed in the last years make use of special indicator functions that explicitly define the optimization goal—independent from the algorithm itself. That is an advantage compared to earlier algorithms where user preferences were incorporated in the algorithms implicitly.

The hypervolume indicator, first introduced by Zitzler et al. as the 'size of the space covered' [13] and also known as the $\mathcal{S}$-metric [1, 7], is used in many cases as the underlying indicator function. Up to now, it is—together with its weighted version of [11]—the only known indicator that is compliant with the concept of Pareto-dominance, i.e., whenever a set of solutions dominates another set, its hypervolume indicator value is higher than the one of the latter. This is the main reason why most of the recently proposed indicator based algorithms like IBEA [12], SMS-EMOA [1], or the multiobjective version of CMA-ES [6] use the hypervolume as the underlying indicator—although its calculation time is exponential in the number of objectives. It was shown experimentally even for a higher number of objectives that hypervolume-based algorithms outperform standard MOEAs [10]. A theoretical understanding why hypervolume-based algorithms outperform their Pareto-dominance based counterparts is still missing.

This paper is a first step towards a general explanation why hypervolume-based algorithms perform better on the known test problems than other state-of-the-art algorithms. Our aim is to gain insights into the optimization process of hypervolume-based algorithms by carrying out rigorous running time analyses. Besides very general non-convergence results on steady-state MOEAs by Zitzler et al. [14], there are no results on the runtime behavior of indicator based evolutionary algorithms known so far. This paper achieves the first results of this kind. Comparisons to former running time analysis results of non-hypervolume-based algorithms allow first conclusions that and when hypervolume-based algorithms are preferable to other algorithms. Within this paper, we consider two important parts of the optimization process. First, we examine how hypervolume-based evolutionary algorithms may approach the Pareto optimal set (Section 3). By considering the function LOTZ, we point out how the population moves to the Pareto front. Second, we examine in Section 4 how the hypervolume indicator helps to spread the individuals of a population over a large Pareto front such that a good approximation of the Pareto optimal set can be achieved. In the following section, we provide the basis for our analyses to follow.

2 The Hypervolume Indicator and Hypervolume-Based Algorithms

Our aim is to analyze hypervolume-based algorithms for multi-objective optimization problems. Without loss of generality, we assume that k objective functions $f = (f_1, \ldots, f_k)$ that map solutions $x \in X$ from the decision space X to an objective vector $f(x) = (f_1(x), \ldots, f_k(x)) \subseteq \mathbb{R}^k$ have to be maximized. Throughout this study, we assume that X is the set of binary strings of length n. Instead of optimizing the weak Pareto-dominance relation $\succeq := \{(x, y) \mid x, y \in X \wedge \forall 1 \leq i \leq k \colon f_i(x) \geq f_i(y)\}$, i.e., finding its maximal elements also called Pareto optimal solutions[1], the goal for hypervolume-based algorithms is to maximize the hypervolume indicator I_H. The hypervolume indicator $I_H(A)$ of a solution set $A \subseteq X$ can be defined as the hypervolume of the space that is dominated by the set A and is bounded by a reference point $r = (r_1, \ldots, r_k) \in \mathbb{R}^k$:

$$I_H(A) = \lambda\left(\bigcup_{a \in A} [f_1(a), r_1] \times [f_2(a), r_2] \times \cdots \times [f_k(a), r_k]\right)$$

where $\lambda(S)$ is the Lebesgue measure of a set S and $[f_1(a), r_1] \times [f_2(a), r_2] \times \cdots \times [f_k(a), r_k]$ is the k-dimensional hypercuboid consisting of all points that are weakly dominated by the point a but not weakly dominated by the reference point.

Note that the hypervolume indicator is Pareto-dominance compliant, i.e., whenever a solution set $A \subseteq X$ is strictly better than a set $B \subseteq X$ with respect to the weak Pareto-dominance relation[2] the hypervolume of A is also strictly better than the one for B ($I_H(A) > I_H(B)$). Therefore, the objective vectors of a set $X^* \subseteq X$ that maximizes the hypervolume indicator cover the Pareto front entirely [2].

[1] Usually, an approximation of the so-called Pareto front is sought: the set of objective vectors the preimages of which are Pareto optimal is usually smaller than the Pareto optimal set itself.

[2] We say that a set of solutions A is strictly better than another set B iff (for maximization) $A \succeq B \wedge B \not\succeq A$ where A is dominating B ($A \succeq B$) iff $\forall b \in B : \exists a \in A : a \succeq b$.

Fixing the maximal number μ of solutions in an evolutionary algorithm $\mathcal{A}$, the goal of maximizing the hypervolume indicator changes to finding a set of μ solutions that have the maximal hypervolume indicator value among all sets of μ solutions. The time until such a solution set is found for the first time is referred to as the optimization time of $\mathcal{A}$; its expectation is denoted by the term *expected optimization time*.

Several evolutionary algorithms to optimize the hypervolume have been proposed in the literature [1, 6, 11, 12]. Most of them use the same $(\mu + \lambda)$-selection scheme which will be also investigated in the remainder of the paper. The population P of the next generation with $|P| = \mu$ is computed from the set P' of solutions that is the union of the previous population and the λ generated offsprings in the following way: after a non-dominated sorting of P' [4], the non-dominated fronts are, starting with the best front, completely inserted into the new population P until the size of P is at least μ. For the first front F the inclusion of which yields a population size larger than μ, the solutions x in this front with the smallest indicator loss $d(x) := I_H(F) - I_H(F \setminus \{x\})$ are successively removed from the new population where the indicator loss is recalculated every time a solution is removed.

The algorithm $(\mu + 1)$-SIBEA, we investigate in the following, is based on the Simple Indicator-Based Evolutionary Algorithm (SIBEA) proposed in [11] that also uses the above mentioned selection scheme. For our theoretical investigations, we consider a simplified version of SIBEA (see Algorithm 1). It uses a population P of size μ and produces in each iteration one single offspring x. By removing the individual with the smallest hypervolume loss from $P \cup \{x\}$, the new parent population is obtained. The omission of the non-dominated sorting step is not crucial for our obtained results, i.e., all running time bounds are the same than with the sorting. Only dominated points are handled differently: with the original selection scheme, always the worst point on the worst front is deleted, whereas in our version, any dominated point is deleted with the same probability.

Algorithm 1. $(\mu + 1)$-SIBEA

```
Parameters: population size $\mu$

Step 1 (Initialization):
Generate an initial (multi)-set of decision vectors $P \subseteq \{0,1\}^n$ of size $\mu$ uniformly at random.
Step 2 (Repeat):
    • Select an element $x$ from $P$ uniformly at random. Flip each bit of $x$ with probability $1/n$ to
      obtain an offspring $x'$. Set $P' := P \cup \{x'\}$.
    • For each solution $x \in P'$ determine the hypervolume loss $d(x)$ if it is removed from $P'$,
      i.e., $d(x) := I_H(P') - I_H(P' \setminus \{x\})$.
    • Choose an element $z \in P'$ with smallest loss in $P'$ uniformly at random, i.e., $z =$
      $\mathrm{argmin}_{x \in P}\, d(x)$ and set $P := P' \setminus \{z\}$.
```

The goal of the next sections is to analyze the runtime behavior of $(\mu + 1)$-SIBEA on some example functions. These analyses point out some basic concepts how the algorithm can make progress during the optimization process. Additionally, it gives insights how a good spread over the whole Pareto front can be achieved using the hypervolume indicator.

3 Exploring a Small Pareto Front

In this section, we examine the well-known bi-objective problem LOTZ with a Pareto front of size $n+1$ and show that the expected optimization time of the $(\mu+1)$-SIBEA is $O(\mu n^2)$ if μ is large enough to find all optima, i.e., $\mu \geq n+1$.

LOTZ was first investigated in [9] and has been considered in several previous studies concerning the running time analysis of MOEAs. It is defined as LOTZ: $\{0,1\}^n \to \mathbb{N}^2$ with

$$f_1(x) = \mathrm{LO}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j \quad \text{and} \quad f_2(x) = \mathrm{TZ}(x) = \sum_{i=1}^{n} \prod_{j=i}^{n} (1-x_j).$$

Without loss of generality, we fix the reference point for computing the hypervolume to $(-1,-1)$. All results of this section still hold as long as the reference point (r_1, r_2) is chosen such that r_1 and r_2 are negative.

Lemma 1. *The expected time until the* $(\mu+1)$*-SIBEA has obtained for the first time a Pareto optimal solution of* LOTZ *is* $O(\mu n^2)$.

Proof. Throughout this proof, we consider the situation where no Pareto optimal search point belongs to the current population P. Let $\{x_1, x_2, \ldots, x_k\} \subseteq P$ be the set of individuals that are not dominated by any other individual in P. Denote by H the hypervolume covered by these points. Without loss of generality, we assume that $\mathrm{LO}(x_i) \leq \mathrm{LO}(x_{i+1})$, $1 \leq i \leq k-1$ holds which also implies $\mathrm{TZ}(x_i) \geq \mathrm{TZ}(x_{i+1})$, $1 \leq i \leq k-1$, as the k individuals do not dominate each other.

Let $X_1 = \mathrm{LO}(x_1)+1$, $X_i = \mathrm{LO}(x_i) - \mathrm{LO}(x_{i-1})$, $2 \leq i \leq k$ and denote by $X_{\max} = \sum_{i=1}^{k} X_i$ the maximum LO-value with respect to the reference point $(-1,-1)$. Similar, define $Y_1 = \mathrm{TZ}(x_k)+1$ and $Y_i = \mathrm{TZ}(x_{k-i}) - \mathrm{TZ}(x_{k-i+1})$, $2 \leq i \leq k$, and denote by $Y_{\max} = \sum_{i=1}^{k} Y_i$ the maximum TZ-value with respect to the reference point $(-1,-1)$.

Considering one single solution x_i of the k non-dominated solutions of P, we study how the hypervolume can increase. Flipping the single bit which increases its LO-value increases the hypervolume by at least Y_{k-i+1}. Flipping the single bit which increases its TZ-value increases the hypervolume by at least X_i. We call all these 1-bit flips applied to one of the k individuals good. Each of these $2k$ good operations happens with probability $\frac{1}{\mu} \cdot \frac{1}{n} \cdot (1-1/n)^{n-1} \geq \frac{1}{e\mu n}$ in the next step.

Note, that each good operation is accepted as it leads to a population with a larger hypervolume. The total increase of all good operations with respect to the current hypervolume H is at least $X_{\max} + Y_{\max} \geq \sqrt{X_{\max} \cdot Y_{\max}} \geq \sqrt{H}$.

Choosing one of theses $2k$ good operations uniformly at random, the expected increase of the hypervolume is at least $\sqrt{H}/(2k)$. Hence, the expected number of good operations needed to increase the hypervolume by $\sqrt{H}$ is upper bounded by $2k$. Using Markov's inequality, the probability of having at least $4k$ operations to achieve this goal is upper bounded by $1/2$. Hence, with probability at least $1/2$ a phase containing $4k$ good operations is successful, i.e., increases the hypervolume by $\sqrt{H}$ with probability at least $1/2$. This implies that an expected number of 2 of these phases carrying out $4k$ such good operations each is enough to increase the hypervolume by $\sqrt{H}$.

Considering all good 1-bit flips together, the probability of carrying out one good operation in the next step of the algorithm is at least $\frac{2k}{e\mu n}$. Hence, the expected waiting time for a good operation is $O(\mu n/(2k))$ and the expected waiting time for increasing the hypervolume by at least $\sqrt{H}$ is therefore upper bounded by $O(\frac{\mu n}{2k} \cdot 2 \cdot 4k) = O(\mu n)$.

It remains to show that $O(n)$ successive increases of the hypervolume by its square-root fraction suffice to reach the maximum hypervolume of $O(n^2)$. Let $h(t)$ be the hypervolume of the current solutions after t increases by at least $\sqrt{h(t)}$. Then, $h(t+1) \geq h(t) + \sqrt{h(t)}$. We want to prove by induction that $h(t) \geq t^2/5$. The induction basis case holds trivially since $h(0) \geq 0$. In general,

$$\begin{aligned} h(t) \geq h(t-1) + \sqrt{h(t-1)} &\geq \frac{(t-1)^2}{5} + \frac{t-1}{\sqrt{5}} \\ &\geq \frac{t^2}{5} + t\left(\frac{1}{\sqrt{5}} - \frac{2}{5}\right) - \left(\frac{1}{\sqrt{5}} - \frac{1}{5}\right) \geq \frac{t^2}{5}. \end{aligned}$$

Therefore, the expected number of iterations for the situation where no solution of the current population is Pareto optimal is upper bounded by $O(\mu n^2)$. □

Theorem 2. *Choosing $\mu \geq n+1$, the expected optimization time of the $(\mu+1)$-SIBEA on* LOTZ *is $O(\mu n^2)$.*

Proof. Using Lemma 1, the expected time until a first Pareto optimal solution has been obtained is $O(\mu n^2)$. Due to the hypervolume-based selection and the fact that at most $n+1$ solutions are mutually non-dominated in LOTZ [9], a Pareto optimal solution that has been found with $(\mu+1)$-SIBEA will stay from that moment in the population or another solution mapping to the same objective vector will enter the population. Increasing the number of Pareto optimal solutions in the population increases the hypervolume indicator, i.e., the highest hypervolume value is achieved if and only if the entire Pareto front is found. Therefore, there is at least one solution in the population which has a Hamming neighbor that is Pareto optimal and not contained in the current population—unless the whole Pareto optimal set is already found. Hence, the expected waiting time for increasing the number of Pareto optimal solutions in the population is $O(\mu n)$. Having reached a Pareto optimal solution for the first time at most n additional Pareto optimal solutions have to be produced which implies that the expected time to achieve a population including all Pareto optimal solutions is $O(\mu n^2)$. □

4 Approximating a Large Pareto Front

The goal of this section is to examine how the hypervolume indicator helps to achieve a good spread over a larger Pareto front. In the case of a large Pareto front, we are interested in the time until an algorithm has achieved a good approximation of the Pareto front. We are considering the multiplicative ε-dominance relation [8] to measure the quality of an approximation. Let $\varepsilon \in \mathbb{R}^+$ be a positive real number. We define that an objective vector u *ε-dominates* v, denoted by $u \succeq_\varepsilon v$, precisely if $(1+\varepsilon) \cdot u_i \geq v_i$ for all $i \in \{1, \ldots, m\}$. An evolutionary algorithm has achieved an ε-approximation for a given problem if there exists for each objective vector v in the objective space a

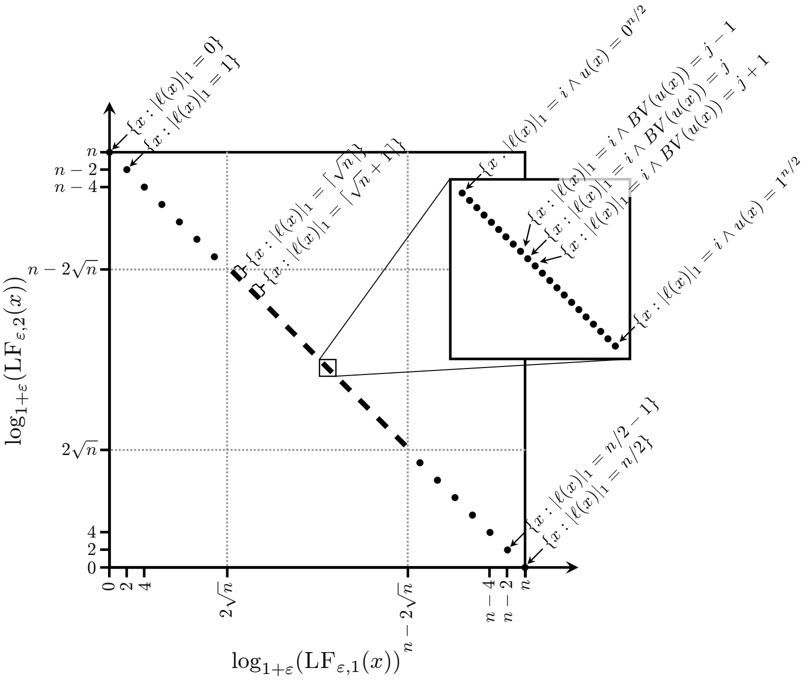


Fig. 1. Illustration of the objective space of LF_ε. In addition, the corresponding solutions in decision space are annotated as well. It is important to note that instead of the original objective values the logarithms of the objective values are plotted for clarity.

solution with objective vector u in the population such that $u \succeq_\varepsilon v$. In the following, we present for each choice of ε a function where the Pareto-dominance based algorithm Global SEMO [3] cannot obtain an ε-approximation while $(\mu+1)$-SIBEA is able to achieve this goal in expected polynomial time.

We consider the bi-objective problem LF_ε (large front) introduced in [5] which is parametrized by the value ε coming from the definition of ε-dominance. Without loss of generality, we assume that n is even, i.e., each decision vector consists of an even number of bits. We denote the lower half of a decision vector $x = (x_1, \dots, x_n)$ by $\ell(x) = (x_1, \dots, x_{n/2})$ and its upper half by $u(x) = (x_{n/2+1}, \dots, x_n)$. Furthermore, we denote the length of a bit-string x by $|x|$, the number of its 1-bits by $|x|_1$, the number of its 0-bits by $|x|_0$, and its complement by $\overline{x}$. In addition, we define the function

$$\mathrm{BV}(x) := \sum_{i=1}^{|x|} 2^{|x|-i} \cdot x_i$$

which interprets a bit-string x as the encoded natural number with respect to the binary numeral system. We consider the function $\mathrm{LF}_\varepsilon \colon \{0,1\}^n \to \mathbb{R}^2$ (large front), for a given $\varepsilon \in \mathbb{R}^+$, defined as

$$f_1(x) = \mathrm{LF}_{\varepsilon,1}(x) := \begin{cases} (1+\varepsilon)^{2\cdot|\ell(x)|_1 + 2^{-n/2}\cdot \mathrm{BV}(u(x))} & \min\{|\ell(x)|_0, |\ell(x)|_1\} \geq \sqrt{n} \\ (1+\varepsilon)^{2\cdot|\ell(x)|_1} & \text{otherwise,} \end{cases}$$

$$f_2(x) = \mathrm{LF}_{\varepsilon,2}(x) := \begin{cases} (1+\varepsilon)^{2\cdot|\ell(x)|_0 + 2^{-n/2}\cdot \mathrm{BV}(\overline{u(x)})} & \min\{|\ell(x)|_0, |\ell(x)|_1\} \geq \sqrt{n} \\ (1+\varepsilon)^{2\cdot|\ell(x)|_0} & \text{otherwise.} \end{cases}$$

The function $\mathrm{LF}_\varepsilon(x)$ is illustrated in Figure 1. In the following proofs it will sometimes be helpful to use the following equivalent formulation of $f_2(x)$:

$$\mathrm{LF}_{\varepsilon,2}(x) = \begin{cases} (1+\varepsilon)^{n-2|\ell(x)|_1 + 1 - 2^{-n/2}\mathrm{BV}(u(x)) - 2^{-n/2}} & \min\{|\ell(x)|_0, |\ell(x)|_1\} \geq \sqrt{n} \\ (1+\varepsilon)^{n-2|\ell(x)|_1} & \text{otherwise.} \end{cases}$$

It has been shown in [5] that Global SEMO needs with probability exponentially close to 1 a number of steps that is exponential in the number of bits to achieve an ε-approximation of LF. The reason for this negative result is that the population of Global SEMO becomes exponentially large before obtaining an ε-approximation. On the other hand it has been pointed out in this paper that the use of ε-dominance with the choice of ε as used for the definition of LF achieves an ε-approximation in expected polynomial time.

In the following, we show that this goal can also be achieved by using the $(\mu+1)$-SIBEA with a population of reasonable size. Our result holds for each $\varepsilon \in \mathbb{R}^+$—in contrast to the usage of ε-dominance based algorithms examined in [5] where the exact knowledge of ε is necessary to achieve a good approximation.

Let the reference point for computing the hypervolume be $((1+\varepsilon)^{-1}, (1+\varepsilon)^{-1})$ corresponding to the point $(-1,-1)$ in the plot of Figure 1. Note that the following results also hold for any reference point (r_1, r_2) with $r_1, r_2 \leq (1+\varepsilon)^{-1}$.

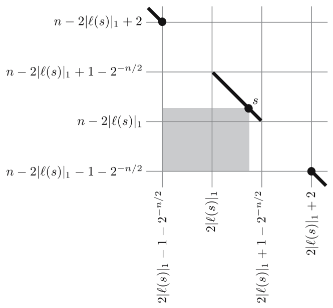
Theorem 3. *Choosing $\mu \geq n/2 + 3$, the expected time until $(\mu+1)$-SIBEA has achieved an ε-approximation of* LF_ε *is $O(\mu n \log n)$.*

To prove Theorem 3 we have to show that if no other solution x with $|\ell(x)|_1 = |\ell(s)|_1$ is contained in the population, the $(\mu+1)$-SIBEA will not remove the solution s from the population:

Lemma 4. *When optimizing* LF_ε*, the $(\mu+1)$-SIBEA with $\mu \geq n/2+3$ will not remove a solution s with $\{x \in P : |\ell(x)|_1 = |\ell(s)|_1\} = \{s\}$ from the population P.*

Proof. If $\{x \in P : |\ell(x)|_1 = k\} = \{s\}$ for some k, such a solution s will be called sole. To show the lemma, it suffices to prove that sole solutions are not removed from the population. Let $d(x) = I_H(P') - I_H(P' \setminus \{x\})$ be the hypervolume loss of a solution x in the population P' of $(\mu+1)$-SIBEA and let s be a sole solution. We will show that there is always another solution z with $d(z) < d(s)$ which proves the lemma due to the selection step of $(\mu+1)$-SIBEA. To this end, we first calculate a lower bound for $d(s)$ and then upper bound $d(z)$. The small sketches to the right of the volume calculations in this proof use the same double-logarithmic axes as Figure 1. If $\min\{|\ell(s)|_0, |\ell(s)|_1\} \geq \sqrt{n}$, then (we can ignore the $-2^{-n/2}$ in the exponent of the first subtrahend)

$$\begin{aligned}
d(s) >& \big[(1+\varepsilon)^{2|\ell(s)|_1+2^{-n/2}\mathrm{BV}(u(s))} - (1+\varepsilon)^{2|\ell(s)|_1-1}\big]\\
&\cdot \big[(1+\varepsilon)^{n-2|\ell(s)|_1+1-2^{-n/2}\mathrm{BV}(u(s))-2^{-n/2}}\\
&\quad - (1+\varepsilon)^{n-2|\ell(s)|_1-1-2^{-n/2}}\big]\\
=&(1+\varepsilon)^{n+1-2^{-n/2}} + (1+\varepsilon)^{n-2-2^{-n/2}}\\
&- (1+\varepsilon)^{n-2^{-n/2}\mathrm{BV}(u(s))-2^{-n/2}}\\
&- (1+\varepsilon)^{n-1+2^{-n/2}\mathrm{BV}(u(s))-2^{-n/2}}\\
\geq&(1+\varepsilon)^{n+1-2^{-n/2}} - (1+\varepsilon)^{n-2^{-n/2}}\\
&- (1+\varepsilon)^{n-1-2^{-n/2}} + (1+\varepsilon)^{n-2-2^{-n/2}}.
\end{aligned}$$



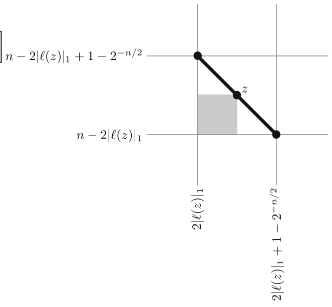
$n-2|\ell(s)|_1+2$
$n-2|\ell(s)|_1+1-2^{-n/2}$
s
$n-2|\ell(s)|_1$
$n-2|\ell(s)|_1-1-2^{-n/2}$
$2|\ell(s)|_1-1-2^{-n/2}$
$2|\ell(s)|_1$
$2|\ell(s)|_1+1-2^{-n/2}$
$2|\ell(s)|_1+2$

where the last inequality stems from the fact that

$$\max_{0\leq\Delta\leq1}(1+\varepsilon)^{\Delta}+(1+\varepsilon)^{1-\Delta} = (1+\varepsilon)^0+(1+\varepsilon)^1.$$

It remains to prove the existence of a solution z with $d(z) < d(s)$. If there is a solution z with $\min\{|\ell(z)|_0, |\ell(z)|_1\} < \sqrt{n}$ and $|\{x \in P : |\ell(x)|_1 = |\ell(z)|_1\}| \geq 2$, then $d(z) = 0$ and the lemma is proven. If there is a k with $|\{x \in P : |\ell(x)|_1 = k\}| > 2$, then there is a solution z with $|\ell(z)|_1 = k$ and

$$\begin{aligned}
d(z) \leq& \big[(1+\varepsilon)^{2|\ell(z)|_1+2^{-n/2}\mathrm{BV}(u(z))} - (1+\varepsilon)^{2|\ell(z)|_1}\big]\\
&\cdot \big[(1+\varepsilon)^{n-2|\ell(z)|_1+1-2^{-n/2}\mathrm{BV}(u(z))-2^{-n/2}}\\
&\quad - (1+\varepsilon)^{n-2|\ell(z)|_1}\big]\\
<&(1+\varepsilon)^{n+1} - (1+\varepsilon)^{n+2^{-n/2}\mathrm{BV}(u(z))}\\
&+ (1+\varepsilon)^{n} - (1+\varepsilon)^{n+1-2^{-n/2}\mathrm{BV}(u(z))}\\
\leq&(1+\varepsilon)^{n+1} + (1+\varepsilon)^{n} - 2(1+\varepsilon)^{n+1/2}.
\end{aligned}$$



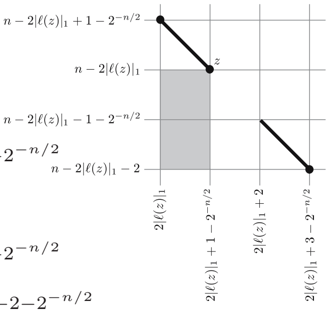
where the last inequality holds since $\operatorname{argmin}_{0\leq\Delta\leq1}(1+\varepsilon)^{\Delta}+(1+\varepsilon)^{1-\Delta} = \frac{1}{2}$. Comparing this upper bound for $d(z)$ and the above lower bound for $d(s)$ yields $d(z) < d(s)$ for all $\varepsilon > 0$ and n.

It remains to examine the case where there is neither a k with $\min\{n-k, k\} < \sqrt{n}$ and $|\{x \in P : |\ell(x)|_1 = k\}| \geq 2$ nor a k with $|\{x \in P : |\ell(x)|_1 = k\}| > 2$. As there are only $n/2+1$ possible values for k, but at least $\mu+1 \geq n/2+4$ solutions in P', by the pigeonhole principle there must be a k with

$$\begin{gathered}
\min\{n-k, k\} > \lceil\sqrt{n}\rceil,\\
|\{x \in P : |\ell(x)|_1 = k\}| \geq 2,\\
|\{x \in P : |\ell(x)|_1 = k+1\}| \geq 1.
\end{gathered}$$

Let z be a solution with $|\ell(z)|_1 = k$, then

$$\begin{aligned}
d(z) \leq & \Big[(1+\varepsilon)^{2|\ell(z)|_1+2^{-n/2}\mathrm{BV}(u(z))} - (1+\varepsilon)^{2|\ell(z)|_1}\Big] \\
& \cdot \Big[(1+\varepsilon)^{n-2|\ell(z)|_1+1-2^{-n/2}\mathrm{BV}(u(z))-2^{-n/2}} \\
& \quad - (1+\varepsilon)^{n-2|\ell(z)|_1-2}\Big] \\
= & (1+\varepsilon)^{n+1-2^{-n/2}} - (1+\varepsilon)^{n+1-2^{-n/2}\mathrm{BV}(u(z))-2^{-n/2}} \\
& - (1+\varepsilon)^{n-2+2^{-n/2}\mathrm{BV}(u(z))} + (1+\varepsilon)^{n-2} \\
< & (1+\varepsilon)^{n+1-2^{-n/2}} - (1+\varepsilon)^{n+1-2^{-n/2}\mathrm{BV}(u(z))-2^{-n/2}} \\
& - (1+\varepsilon)^{n-2+2^{-n/2}\mathrm{BV}(u(z))-2^{-n/2}} + (1+\varepsilon)^{n-2-2^{-n/2}} \\
\leq & (1+\varepsilon)^{n+1-2^{-n/2}} - (1+\varepsilon)^{n-2^{-n/2}} - (1+\varepsilon)^{n-1-2^{-n/2}} + (1+\varepsilon)^{n-2-2^{-n/2}}.
\end{aligned}$$


$n-2|\ell(z)|_1+1-2^{-n/2}$
z
$n-2|\ell(z)|_1$
$n-2|\ell(z)|_1-1-2^{-n/2}$
$n-2|\ell(z)|_1-2$
$2|\ell(z)|_1$
$2|\ell(z)|_1+1-2^{-n/2}$
$2|\ell(z)|_1+2$
$2|\ell(z)|_1+3-2^{-n/2}$

where the last inequality comes from $\mathrm{argmin}_{0\leq\Delta\leq 1}(1+\varepsilon)^{3-\Delta}+(1+\varepsilon)^{\Delta}=1$. This shows $d(z)<d(s)$ and finally proves the lemma. □

Proof of Theorem 3. An ε-approximation of the Pareto front has been achieved if and only if the population includes for each $k\in\{0,\ldots n/2\}$ a solution x with $|\ell(x)|_1=k$ (see [5]). Denote the set of covered $|\cdot|_1$ values by $A:=\{|l(x)|_1 \mid x\in P\}$ and the set of uncovered $|\cdot|_1$ values by $B:=\{0,\ldots,n/2\}\setminus A$. Due to the previous lemma, we know that during the optimization process elements that are added to A are never removed and follow the ideas given in [5].

As long as $A\neq\{0,\ldots,n/2\}$, there exists an $a\in A$ and a $b\in B$ with $b=a-1$ or $b=a+1$. Let $x\in P$ be the individual with $|l(x)|_1=a$. The probability to choose x in the next step and flip exactly one proper bit to obtain a decision vector y with $|l(y)|_1=b$ is at least $\frac{1}{\mu}\cdot\frac{\min\{b+1,n/2-b+1\}}{en}\geq\frac{\min\{b,n/2-b\}+1}{\mu en}$.

Summing up over the different values that b can attain, we get a maximum waiting time of $\mu en\cdot\sum_{b=0}^{n/2}\frac{1}{\min\{b,n/2-b\}+1}\leq 2\mu en\cdot\sum_{b=1}^{n/4+1}\frac{1}{b}=\mathcal{O}(\mu n\log n)$ until solutions with all possible values of b are contained in the population which completes the proof. □

5 Conclusions

Indicator-based evolutionary algorithms have been shown to be very successful for dealing with multiobjective optimization. With this paper, we have taken a first step in understanding these algorithms using the hypervolume indicator by rigorous running time analysis. Considering the function LOTZ, we have pointed out how the progress of such algorithms towards the Pareto front can be analyzed. Later on, we have shown that the hypervolume indicator is provable helpful for approximating large Pareto fronts.

Acknowledgments

The authors would like to thank Johannes Bader, Tim Hohm, and Tamara Ulrich for valuable discussions. Dimo Brockhoff has been supported by the Swiss National Science Foundation (SNF) under grant 112079.

Bibliography

[1] Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal on Operational Research 181, 1653–1669 (2007)

[2] Fleischer, M.: The measure of Pareto optima. Applications to multi-objective metaheuristics. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 519–533. Springer, Heidelberg (2003)

[3] Giel, O.: Expected Runtimes of a Simple Multi-objective Evolutionary Algorithm. In: Congress on Evolutionary Computation (CEC 2003), pp. 1918–1925. IEEE Press, Los Alamitos (2003)

[4] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)

[5] Horoba, C., Neumann, F.: Benefits and drawbacks for the use of ε-dominance in evolutionary multi-objective optimization. In: Genetic and Evolutionary Computation Conference (GECCO 2008) (to appear, 2008); Also available as Technical Report CI-248/08, SFB 531, TU Dortmund

[6] Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-objective Optimization. Evolutionary Computation 15(1), 1–28 (2007)

[7] Knowles, J., Corne, D.: Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors. IEEE Transactions on Evolutionary Computation 7(2), 100–116 (2003)

[8] Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. Evolutionary Computation 10(3), 263–282 (2002)

[9] Laumanns, M., Thiele, L., Zitzler, E.: Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions. IEEE Transactions on Evolutionary Computation 8(2), 170–182 (2004)

[10] Wagner, T., Beume, N., Naujoks, B.: Pareto-, Aggregation-, and Indicator-based Methods in Many-objective Optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 742–756. Springer, Heidelberg (2007)

[11] Zitzler, E., Brockhoff, D., Thiele, L.: The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)

[12] Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN VIII 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)

[13] Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)

[14] Zitzler, E., Thiele, L., Bader, J.: On Set-Based Multiobjective Optimization. Technical Report 300, Computer Engineering and Networks Laboratory, ETH Zurich (February 2008)

Solving Three-Objective Optimization Problems Using a New Hybrid Cellular Genetic Algorithm

J.J. Durillo, A.J. Nebro, F. Luna, and E. Alba

Department of Computer Science, University of Málaga, Spain
{durillo,antonio,flv,eat}@lcc.uma.es

Abstract. In this work we present a new hybrid cellular genetic algorithm. We take MOCell as starting point, a multi-objective cellular genetic algorithm, and, instead of using the typical genetic crossover and mutation operators, they are replaced by the reproductive operators used in differential evolution. An external archive is used to store the nondominated solutions found during the search process and the SPEA2 density estimator is applied when the archive becomes full. We evaluate the resulting hybrid algorithm using a benchmark composed of three-objective test problems, and we compare the results with several state of the art multi-objective metaheuristics. The obtained results show that our proposal outperforms the other algorithms according to the two considered quality indicators.

1 Introduction

Multi-objective optimization refers to optimizing problems whose formulation involves two or more objectives, which are known as multi-objective optimization problems (MOPs). The solution to these kinds of problem uses not to be a single one; instead, a set of *nondominated solutions* has to be found. Each solution in this set is said to be a *Pareto optimum*, and when they are plotted in the objective space they are collectively known as the *Pareto front*.

In the last few years evolutionary algorithms (EAs) have become very popular tools for solving MOPs since they are capable of obtaining the Pareto front in a single run. As a consequence, many multi-objective EAs have appeared in recent years, and the most well-known metaheuristics, such as NSGA-II [1], SPEA2 [2], PAES [3], and many others [4][5], belong to this family of techniques. Most of these algorithms are genetic algorithms (GA), a subclass into EAs.

Our starting point is MOCell [6], a multi-objective cellular GA (cGA) that is characterized by the use of an external archive to store the non-dominated solutions found during the search and a feedback mechanism in which solutions from this archive randomly replaces existing individuals in the population after each iteration. In order to manage the insertion of solutions in the archive with the goal of obtaining a diverse set, MOCell includes a density estimator based on the crowding distance of NSGA-II [1]. This measure is also used to remove solutions from the archive when it becomes full. MOCell has proven to be very effective in solving bi-objective MOPs; in particular, it provides Pareto fronts with a remarkable uniformity (spread) of their solutions. However, preliminary experiments have revealed that it has difficulties when dealing with three-objective MOPs (namely, those belonging to the DTLZ problem family [7]).

In our research activity, we paid attention to differential evolution (DE) algorithms [8], another kind of EA. DE has been successfully applied as a single-objective optimizer in continuous search problems within the last few years [9], and there are proposals which adapt it to multi-objective optimization [10,11,12]. In particular, we focused on the Generalized Differential Evolution 3 (GDE3) algorithm [11]. Preliminary experiments with GDE3 showed that it was able to reach solution sets which are very close to the Pareto front when solving some DTLZ problems.

This work is aimed at designing a metaheuristic capable of producing the same satisfactory results in three-objective MOPs as MOCell achieves in bi-objective problems. Our proposal is a new hybrid metaheuristic, called CellDE, which tries to combine the advantages of both MOCell (good diversity in bi-objective MOPs) and GDE3 (good convergence in three-objective MOPs). The idea is to use MOCell as search engine and hybridizing it with DE, by replacing the typical genetic operators of crossover and mutation of GAs by the reproductive mechanism used in DE.

To assess the performance of our algorithm, we have compared it to the techniques it derives, MOCell and GDE3, and to NSGA-II and SPEA2, the reference metaheuristics in the field. We have used a benchmark composed of the three-objective formulation of the MOPs included in the DTLZ and WFG [13] problem families.

The rest of the paper is organized as follows. In Section 2, we give an introduction to cellular GAs and DE. Our proposal is described in Section 3. Section 4 is devoted to analyzing the obtained results in the experiments. Finally, Section 5 includes the conclusions and lines of future work.

2 Cellular GAs and Differential Evolution

GAs work on a set (*population*) of tentative solutions (*individuals*) which undergoes stochastic operators (typically selection, crossover, and mutation) in order to search for better solutions. The form in which this set of solutions is structured yields to different kinds of GAs (see Fig. 1). On the one hand, those algorithms that use a single population (panmixia) of individuals and apply operators to them as a whole; on the other hand, the so-called structured GAs, in which the population is decentralized somehow. Among the many types of structured GAs [14], the distributed and cellular models are two popular variants.

Cellular GAs (cGAs) make use of the concept of (small) *neighborhood* in the sense that one individual can only interact with individuals belonging to its neighborhood in the breeding loop. These neighborhoods are defined among tentative solutions in the algorithm, with no relation to the geographical neighborhood definition in the problem space. The overlapped small neighborhoods of cGAs help in exploring the search space: the induced slow diffusion of solutions

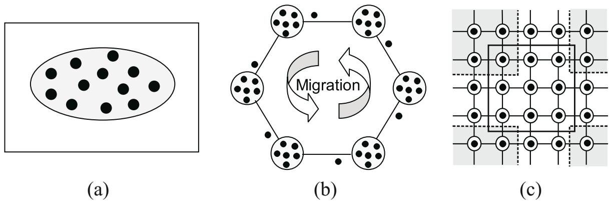


Fig. 1. Panmictic (a), distributed (b), and cellular (c) GAs

through the population provides a kind of exploration (diversification), while exploitation (intensification) takes place inside each neighborhood by genetic operators.

Differential evolution [8] is an evolutionary technique which is gaining popularity in recent years. Like many others EAs, DE uses a population of individuals which are recombined to reach improved solutions. In DE the search process is guided by generating a single offspring by adding a weighted difference vector between two parents to a third parent.

DE works as follows. At each generation G, for each D dimensional solution $\boldsymbol{x}_{i,G}$, $i = 1, 2, \ldots, N$ (N is the population size), a new trial solution $\boldsymbol{u}$ is obtained as it is indicated in Algorithm 1, where CR controls the crossover operation and F is the scaling factor for mutation. Both CR and F remain constant during the execution of the algorithm. After that, the new solution $\boldsymbol{u}_{i,G}$ is compared to the old vector $\boldsymbol{x}_{i,G}$, and the latter is replaced by the former if this one has an equal or better objective value.

3 Outline of CellDE

In this section we describe our proposal. The pseudocode of the algorithm is shown in Algorithm 2. The basic behavior of CellDE is that of a cGA following an asynchronous behavior, in the sense that all the cells are explored sequentially (in synchronous cGAs the cells are explored in parallel). The MOCell version taken as starting point is based on aMOCell3 [6], which is characterized by using an external archive to store the non-dominated solutions found so far

Algorithm 1. Pseudocode of generating a new solution in DE.

```
1: // $r_1$, $r_2$, $r_3 \in \{1, 2, \ldots, N\}$, randomly selected, except mutually different from $i$
2: proc differentialEvolution($i$, $r_1$, $r_2$, $r_3$)
3: $jrand$ =floor($rand_i[0,1) \cdot D$) + 1
4: for ($j = 1; j \leqslant D; j = j + 1$) do
5:     if ($rand_j[0,1) < CR \vee j = j_{rand}$) then
6:         $\boldsymbol{u}_{i[j],G} = \boldsymbol{x}_{r3[j],G} + F \cdot (\boldsymbol{x}_{r1[j],G} - \boldsymbol{x}_{r2[j],G})$
7:     else
8:         $\boldsymbol{u}_{i[j],G} = \boldsymbol{x}_{i[j],G}$
9:     end if
10: end for
11: return $\boldsymbol{u}_{i,G}$
12: end_proc differentialEvolution;
```

Algorithm 2. Pseudocode of CellDE.

```
1: proc stepsUp(CellDE)                    //Algorithm parameters in 'CellDE'
2: population ← randomPopulation()         //Creates a random initial population
3: archive ← createFront()                 //Creates an empty Pareto front
4: while !terminationCondition() do
5:   for individual ← 1 to CellDE.populationSize do
6:     neighborhood←getNeighbors(population,position(individual));
7:     parent1←selection(neighborhood);
8:     parent2←selection(neighborhood);
9:     // parent1 and parent2 may be different
10:    while parent1=parent2 do
11:      parent2←selection(neighborhood);
12:    end while
13:    offspring←differentialEvolution(position(individual), position(individual),
                                       position(parent1), position(parent2));
14:    evaluateFitness(offspring);
15:    insert(position(individual),offspring,population);
16:    addToArchive(individual);
17:  end for
18:  population←replaceIndividuals(population,archive);
19: end while
20: end_proc stepsUp;
```

during the search and a feedback mechanism. The aMOCell3 algorithm was originally engineered using the crowding distance as density estimator to manage the diversity in the approximated Pareto front. As it has been reported in the literature [15], this estimator does not perform well with MOPs having more than two objectives. This leads us to use the density estimator of SPEA2 [2] in CellDE and also in the aMOCell3 algorithm used in this work.

The main difference between CellDE and MOCell (we will refer aMOCell3 as MOCell in the rest of the paper) arises in the creation of new individuals. Instead of using the classical GA operators to generate new individuals, CellDE takes the operator used in DE: three different individuals are chosen and the new offspring solution is obtained based on the differences between them. Please, refer to [6] for a detailed description of the methods that will be used next.

CellDE starts by creating a population of random solutions and an empty Pareto front (lines 2 and 3 in Algorithm 2). Individuals are arranged in a 2-dimensional grid, defining neighborhood structures over the population. For each individual $\boldsymbol{x}_{i,G}$, two different solutions of the neighborhood are selected (lines 7 and 8) which, along with the current individual, are used as the three parents to create the new offspring (line 13). This is a different approach to the one used in DE, where the three parents exclude the current solution; we take this scheme since it allows to enhance the intensification capabilities of the algorithm. The newly generated offspring is evaluated (line 14) and then it replaces the original solution if dominates it, or, if both are non-dominated, it replaces the worst individual in the neighborhood (line 15). After that, the new individual is sent to the archive, where it is checked for its insertion (line 16). Finally, after each generation, a feedback procedure is performed to replace a number of randomly chosen individuals by a number of solutions taken from the archive (line 18).

Table 1. Parameterization (L = individual length)

Parameterization used in NSGA-II [1]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Parameterization used in SPEA2 [2]	
Population Size	100 individuals
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Parameterization used in GDE3 [11]	
Population Size	100 individuals
Recombination	Differential Evolution, $CR = 0.1$, $F = 0.5$
Parameterization used in MOCell (aMOCell3) [6]	
Population Size	100 individuals (10×10)
Neighborhood	1-hop neighbors (8 surrounding solutions)
Selection of Parents	binary tournament + binary tournament
Recombination	simulated binary, $p_c = 0.9$
Mutation	polynomial, $p_m = 1.0/L$
Archive Size	100 individuals
Feedback	20% of the population (20 individuals)
Parameterization used in CellDE	
Population Size	100 individuals (10×10)
Neighborhood	1-hop neighbors (8 surrounding solutions)
Selection of Parents	binary tournament + binary tournament
Recombination	differential evolution, $CR = 0.1$, $F = 0.5$
Archive Size	100 individuals
Feedback	20% of the population (20 individuals)

4 Computational Results

This section is devoted to the evaluation of CellDE. We have chosen several test problems taken from the specialized literature, and, in order to assess how competitive CellDE is, we have compared it to the two reference algorithms in the field, namely NSGA-II and SPEA2, as well as to the base algorithms used for designing CellDE, GDE3 and MOCell. All the algorithms have been implemented in Java using the jMetal framework [16].

The parameter settings used in the experiments are summarized in Table 1. The values are taken from the reference papers where the algorithms are described. The stopping condition in all of them is to evaluate 25000 solutions.

The test problems we have used are the three-objective formulations of the Deb-Thiele-Laumanns-Zitzler (DTLZ) benchmark [7] and the Walking-Fish-Group (WFG) problems [13]. A total number of sixteen MOPs has been used to evaluate the five metaheuristics. For assessing the performance of the algorithms, we have used two Pareto-compliant indicators: hypervolume (HV) [17] and additive epsilon indicator ($I^1_{\epsilon+}$) [18]. The latter is an indicator measuring the convergence of the resulting Pareto fronts, while the former measures both convergence and diversity.

We have made 100 independent runs of each experiment, and we have obtained the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively. Since we are dealing with stochastic algorithms and we want to provide the results with confidence,

Table 2. Median and interquartile range of the (additive) Epsilon (I_ϵ) indicator

	NSGA-II	SPEA2	GDE3	MOCell	CellDE	
Problem	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	
DTLZ1	7.62e-2 $_{7.2e-2}$	4.16e-2 $_{8.4e-3}$	4.80e-2 $_{6.6e-3}$	5.35e-1 $_{5.1e-1}$	3.34e-2 $_{3.3e-3}$	+
DTLZ2	1.24e-1 $_{2.0e-2}$	8.20e-2 $_{9.5e-3}$	1.17e-1 $_{1.7e-2}$	7.99e-2 $_{8.5e-3}$	7.62e-2 $_{8.8e-3}$	+
DTLZ3	4.51e+0 $_{2.7e+0}$	4.73e+0 $_{3.0e+0}$	1.36e+1 $_{5.2e+0}$	1.67e+1 $_{7.8e+0}$	3.55e+0 $_{3.3e+0}$	+
DTLZ4	1.12e-1 $_{2.4e-2}$	7.93e-2 $_{5.6e-1}$	1.08e-1 $_{1.9e-2}$	6.92e-2 $_{1.0e-2}$	6.77e-2 $_{8.6e-3}$	+
DTLZ5	1.07e-2 $_{2.6e-3}$	7.74e-3 $_{1.5e-3}$	5.58e-3 $_{4.8e-4}$	8.08e-3 $_{1.6e-3}$	6.55e-3 $_{1.1e-3}$	+
DTLZ6	8.57e-1 $_{1.3e-1}$	7.82e-1 $_{6.3e-2}$	5.10e-3 $_{5.5e-4}$	1.72e+0 $_{1.5e-1}$	6.00e-3 $_{7.9e-4}$	+
DTLZ7	1.27e-1 $_{4.5e-2}$	9.82e-2 $_{1.2e-2}$	1.20e-1 $_{3.6e-2}$	1.15e-1 $_{3.0e-2}$	8.42e-2 $_{1.6e-2}$	+
WFG1	5.66e-1 $_{6.8e-2}$	6.56e-1 $_{1.1e-1}$	7.76e-1 $_{1.1e-1}$	6.30e-1 $_{1.8e-1}$	1.03e+0 $_{1.5e-1}$	+
WFG2	3.23e-1 $_{6.4e-2}$	2.37e-1 $_{3.4e-2}$	3.02e-1 $_{4.5e-2}$	2.56e-1 $_{3.8e-2}$	2.52e-1 $_{3.9e-2}$	+
WFG3	1.24e-1 $_{3.5e-2}$	9.22e-2 $_{1.7e-2}$	1.08e-1 $_{3.6e-2}$	8.57e-2 $_{1.8e-2}$	1.04e-1 $_{3.0e-2}$	+
WFG4	4.32e-1 $_{7.8e-2}$	3.26e-1 $_{3.8e-2}$	4.21e-1 $_{1.0e-1}$	2.95e-1 $_{4.3e-2}$	3.10e-1 $_{4.1e-2}$	+
WFG5	4.71e-1 $_{7.8e-2}$	3.52e-1 $_{4.6e-2}$	4.34e-1 $_{6.4e-2}$	3.44e-1 $_{4.2e-2}$	3.30e-1 $_{4.7e-2}$	+
WFG6	4.31e-1 $_{6.7e-2}$	3.30e-1 $_{4.9e-2}$	3.94e-1 $_{6.2e-2}$	3.13e-1 $_{4.4e-2}$	2.81e-1 $_{3.6e-2}$	+
WFG7	4.65e-1 $_{8.7e-2}$	3.37e-1 $_{3.9e-2}$	4.57e-1 $_{1.1e-1}$	3.07e-1 $_{3.8e-2}$	2.95e-1 $_{3.7e-2}$	+
WFG8	7.51e-1 $_{9.2e-2}$	6.22e-1 $_{1.4e-1}$	7.56e-1 $_{5.4e-2}$	6.26e-1 $_{1.6e-1}$	6.38e-1 $_{3.3e-2}$	+
WFG9	4.39e-1 $_{7.2e-2}$	3.28e-1 $_{4.2e-2}$	4.25e-1 $_{5.8e-2}$	3.13e-1 $_{4.5e-2}$	3.14e-1 $_{3.7e-2}$	+

Table 3. Non-successful statistical test of the I_ϵ indicator

	NSGA-II	SPEA2	GDE3	MOCell
SPEA2	DTLZ3, DTLZ6			
GDE3	DTLZ2, DTLZ4, DTLZ7, WFG4, WFG5, WFG7, WFG8, WFG9	DTLZ1		
MOCell	DTLZ7	DTLZ2, DTLZ5, WFG1, WFG5, WFG6, WFG8	DTLZ3, DTLZ7	
CellDE	DTLZ3	DTLZ3, WFG2, WFG4, WFG5, WFG8	WFG3	DTLZ2, DTLZ4, WFG2, WFG4, WFG5, WFG7, WFG8, WFG9

the following statistical analysis has been performed in all this work [19]. Firstly, a Kolmogorov-Smirnov test is applied in order to check whether the values of the results follow a normal (gaussian) distribution or not. If the distribution is normal, the Levene test checks for the homogeneity of the variances. If samples have equal variance (positive Levene test), an ANOVA test is done; otherwise a Welch test is performed. For non-gaussian distributions, the non-parametric Kruskal-Wallis test is used to compare the medians of the algorithms. We always consider a confidence level of 95% (i.e., significance level of 5% or p-value under 0.05) in the statistical tests. Successful tests are marked with '+' symbols in the last column in all the tables containing the results; conversely, '-' means that no statistical confidence was found (p-value > 0.05). The best result for each problem has a gray colored background. For the sake of a better understanding of the results, we have also used a clearer grey background to indicate the second best result.

To further analyze the results statistically, we have also included a post-hoc testing phase which allows for a multiple comparison of samples [20]. We have used the `multcompare` function provided by Matlab©. Tables 3 and 5 summarize this comparison by including only those problems for which the differences are not statistically different.

We start by analyzing the results of the I_ϵ indicator, which are included in Table 2. We observe that CellDE obtains the best (lowest) values in eight out of

Table 4. Median and interquartile range of the *HV* indicator

	NSGA-II	SPEA2	GDE3	MOCell	CellDE	
Problem	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	
DTLZ1	7.22e-1 $_{1.0e-1}$	7.69e-1 $_{1.5e-2}$	7.62e-1 $_{6.0e-3}$	0.00e+0 $_{1.0e-1}$	7.86e-1 $_{7.9e-4}$	+
DTLZ2	3.73e-1 $_{8.3e-3}$	4.05e-1 $_{2.6e-3}$	3.74e-1 $_{6.3e-3}$	4.10e-1 $_{2.0e-3}$	4.16e-1 $_{1.3e-3}$	+
DTLZ3	–	–	–	–	–	-
DTLZ4	3.74e-1 $_{7.6e-3}$	3.98e-1 $_{1.9e-1}$	3.71e-1 $_{5.9e-3}$	4.05e-1 $_{1.9e-3}$	4.07e-1 $_{1.4e-3}$	+
DTLZ5	9.28e-2 $_{3.0e-4}$	9.32e-2 $_{1.9e-4}$	9.39e-2 $_{7.0e-5}$	9.33e-2 $_{1.7e-4}$	9.36e-2 $_{6.9e-5}$	+
DTLZ6	–	–	9.49e-2 $_{4.8e-5}$	–	9.46e-2 $_{8.1e-5}$	-
DTLZ7	2.80e-1 $_{6.0e-3}$	2.90e-1 $_{3.5e-3}$	2.92e-1 $_{2.8e-3}$	2.81e-1 $_{7.2e-3}$	3.03e-1 $_{2.4e-3}$	+
WFG1	7.71e-1 $_{5.2e-2}$	6.75e-1 $_{7.4e-2}$	6.42e-1 $_{5.4e-2}$	7.17e-1 $_{1.3e-1}$	5.27e-1 $_{1.1e-1}$	+
WFG2	9.01e-1 $_{4.7e-3}$	9.13e-1 $_{1.9e-3}$	9.05e-1 $_{3.3e-3}$	9.12e-1 $_{1.7e-3}$	9.14e-1 $_{1.8e-3}$	+
WFG3	3.19e-1 $_{2.5e-3}$	3.11e-1 $_{2.7e-3}$	3.23e-1 $_{1.5e-3}$	3.15e-1 $_{1.6e-3}$	3.11e-1 $_{4.6e-3}$	+
WFG4	3.65e-1 $_{8.2e-3}$	3.92e-1 $_{4.8e-3}$	3.52e-1 $_{8.6e-3}$	4.07e-1 $_{2.3e-3}$	3.95e-1 $_{4.4e-3}$	+
WFG5	3.41e-1 $_{9.4e-3}$	3.68e-1 $_{6.9e-3}$	3.55e-1 $_{4.0e-3}$	3.68e-1 $_{4.8e-3}$	3.71e-1 $_{1.9e-3}$	+
WFG6	3.64e-1 $_{1.0e-2}$	3.91e-1 $_{1.4e-2}$	3.81e-1 $_{9.0e-3}$	3.97e-1 $_{1.5e-2}$	4.16e-1 $_{2.6e-3}$	+
WFG7	3.58e-1 $_{1.0e-2}$	3.83e-1 $_{5.5e-3}$	3.63e-1 $_{7.8e-3}$	4.00e-1 $_{3.2e-3}$	4.07e-1 $_{2.5e-3}$	+
WFG8	2.42e-1 $_{6.7e-3}$	2.69e-1 $_{9.2e-3}$	2.40e-1 $_{5.5e-3}$	2.69e-1 $_{7.7e-3}$	2.59e-1 $_{5.1e-3}$	+
WFG9	3.57e-1 $_{7.2e-3}$	3.77e-1 $_{3.8e-3}$	3.61e-1 $_{5.4e-3}$	3.86e-1 $_{5.9e-3}$	3.86e-1 $_{2.7e-3}$	+

Table 5. Non-successful statistical test of the *HV* indicator

	NSGA-II	SPEA2	GDE3	MOCell
SPEA2	DTLZ4, DTLZ6			
GDE3	DTLZ2, DTLZ4	DTLZ1, DTLZ4, DTLZ7		
MOCell	DTLZ6, DTLZ7, WFG8	DTLZ6, WFG1, WFG2, WFG5, WFG6, WFG8		
CellDE		WFG2, WFG3		

the sixteen problems evaluated and the second best results in five cases. MOCell is the second best algorithm (three best results and seven second best values) followed by SPEA2 (best value in two out of the sixteen problems evaluated and the second best value in three other problems). GDE only yields the best values in two problems. NSGA-II is the technique providing the poorest fronts, which confirms the fact that this algorithm has difficulties when solving MOPs having more than two objectives.

Table 3 contains, for each pair of algorithms, the MOPs for which no statistical difference exists (at a confidence level of 95%) according to the I_ϵ indicator. If we focus on CellDE, we can see that the differences in the values in five and eight problems with SPEA2 and MOCell, respectively, are not significant. This means that both SPEA2 and MOCell produce similar Pareto fronts in those problems.

We analyze now the results obtained after applying the *HV* indicator (see Table 4). It can be seen that CellDE clearly outperforms the other algorithms, obtaining the best (highest) values in nine out of the sixteen MOPs evaluated, yielding also the second best values in three other problems. MOCell can be considered as the second most competitive algorithm according to *HV* since, although it reaches the best *HV* value in only a single MOP, it is the second best in eight out of the sixteen problems. GDE3 gets the best value in three MOPs, and the second best value only in one case, while SPEA2 obtains the best value in only one problem and the second best value in two cases. The least algorithm with respect to this indicator is NSGA-II, which only reaches the best value in one problem, yielding also the second best value in another one. As

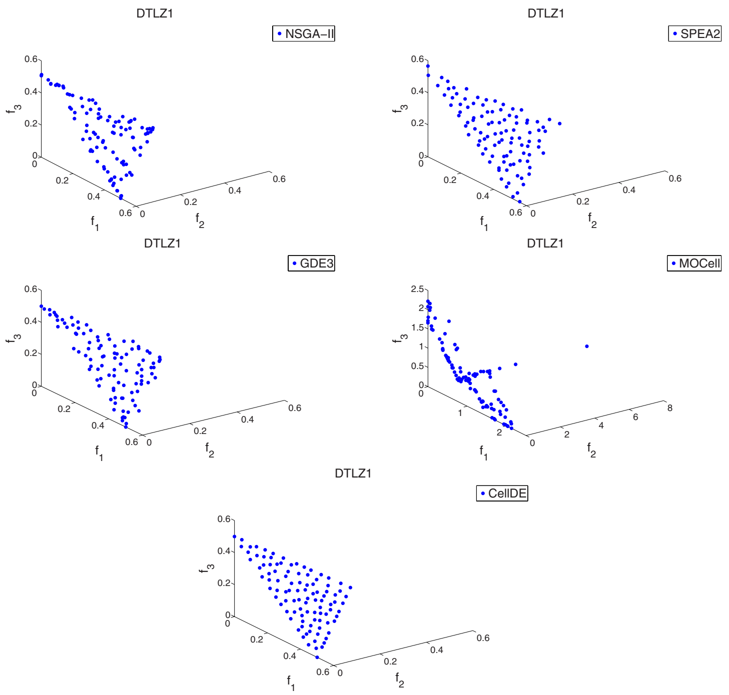
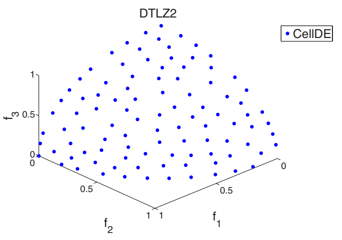


Fig. 2. Front obtained when solving DTLZ1. From left to right, from top to bottom: NSGA-II, SPEA2, GDE3, MOCell, CellDE.

to GDE3 and MOCell, the base algorithms for CellDE, we can state that the search capabilities of the new approach improves significantly those of the two former ones according to HV. We explain now the meaning of the '–' symbol in Table 4. Since the HV indicator is not free from the arbitrary scaling of the objectives, the resulting Pareto fronts of the algorithms have to be normalized. In this normalization process, the nondominated solutions that are outside the limits of the true Pareto front are not considered to compute the HV value because, otherwise, the obtained values would be unreliable.

Table 5 contains, for each pair of algorithms, the MOPs for which no statistical difference appears. The main conclusion that can be drawn from this table is that the differences in the HV values of CellDE with respect to the values of the other four algorithms are significant in all except two MOPs (WFG2 and WFG3 with SPEA2), thus providing our previous claims with statistical support.

To illustrate the search capabilites of CellDE, we include in Fig. 2 the Pareto fronts reached by the different algorithms evaluated when solving problem DTLZ1. We observe that the fronts obtained by CellDE and SPEA2 have a better distribution of solutions than the other ones. Furthermore, in the case of CellDE, all the solutions have converged towards the true Pareto front, while in the SPEA2 front some solutions have not. We include in Fig. 3 the fronts obtained by CellDE for



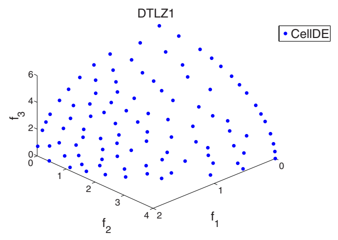


Fig. 3. Fronts obtained by CellDE when solving DTLZ2 (left) and WFG7 (right)

problems DTLZ2 and WFG7, where a uniform distribution of the solutions can be observed.

5 Conclusions and Future Work

In this work we have proposed a new algorithm called CellDE, which hybridizes the behavior of a cellular GA with a DE algorithm. It has been evaluated using a benchmark composed of sixteen three-objective optimization problems.

To assess how competitive CellDE is, we have compared it to four state-of-the-art algorithms, NSGA-II, SPEA2, MOCell, and GDE3, being the last two ones the starting point to design our algorithm. The obtained results show that CellDE clearly outperforms the other techniques according to the parameter settings, problems, and quality indicators used.

A study of the behavior of CellDE when applied to problems having more than three objectives is a matter of future work.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Education and Science and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project). Juan J. Durillo is supported by grant AP-2006-03349 from the Spanish Ministry of Education and Science.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
2. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2001)
3. Knowles, J., Corne, D.: The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 9–105. IEEE Press, Piscataway (1999)

4. Coello, C., Van Veldhuizen, D., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. In: Genetic Algorithms and Evolutionary Computation, 2nd edn. Kluwer Academic Publishers, Dordrecht (2007)
5. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
6. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: Design issues in a multiobjective cellular genetic algorithm. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 126–140. Springer, Heidelberg (2007)
7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
8. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA (1995)
9. Price, K., Storn, R., Lampinen, J.: Differential Evolution A Practical Approach to Global Optimization. Natural Computing Series. Springer, Berlin (2005)
10. Lampinen, J.: De's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology (2001)
11. Kukkonen, S., Lampinen, J.: GDE3: The third Evolution Step of Generalized Differential Evolution. In: IEEE Congress on Evolutionary Computation (CEC 2005), pp. 443–450 (2005)
12. Hernández-Díaz, A.G., Santana-Quintero, L.V., Coello, C.A.C., Caballero, R., Molina, J.: A new proposal for multi-objective optimization using differential evolution and rough sets theory. In: Conference on Genetic and Evolutionary Computation (GECCO 2006), pp. 675–682 (2006)
13. Huband, S., Hingston, P., Barone, L., While, L.: A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477–506 (2006)
14. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation 6(5), 443–462 (2002)
15. Kukkonen, S., Deb, K.: Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In: 2006 IEEE Congress on Evolutionary Computation (CEC 2006), pp. 1179–1186 (2006)
16. Durillo, J.J., Nebro, A.J., Luna, F., Dorronsoro, B., Alba, E.: jMetal: a java framework for developing multi-objective optimization metaheuristics. Technical Report ITI-2006-10, Dpto. de Lenguajes y Ciencias de la Computación, University of Málaga (2006)
17. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
18. Knowles, J., Thiele, L., Zitzler, E.: A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
19. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
20. Hochberg, Y., Tamhane, A.C.: Multiple Comparison Procedures. Wiley, Chichester (1987)

Runtime Analyses for Using Fairness in Evolutionary Multi-Objective Optimization*

Tobias Friedrich[1], Christian Horoba[2], and Frank Neumann[1]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2] Fakultät für Informatik, LS 2, TU Dortmund, Dortmund, Germany

Abstract. It is widely assumed that evolutionary algorithms for multi-objective optimization problems should use certain mechanisms to achieve a good spread over the Pareto front. In this paper, we examine such mechanisms from a theoretical point of view and analyze simple algorithms incorporating the concept of fairness introduced by Laumanns et al. [7]. This mechanism tries to balance the number of offspring of all individuals in the current population. We rigorously analyze the runtime behavior of different fairness mechanisms and present showcase examples to point out situations where the right mechanism can speed up the optimization process significantly.

1 Introduction

Evolutionary algorithms evolve a set of solutions called the population during the optimization process. In multi-objective optimization one usually does not search for a single optimal solution but a set of solutions representing the possible trade-offs when dealing with conflicting objective functions. Hence, multi-objective evolutionary algorithms (MOEAs) seem to be in a natural way well suited for dealing with these problems.

Many MOEAs give priority to regions in the decision or objective space that have been rarely explored. This leads to the use of fairness in evolutionary multi-objective optimization. The idea behind using fairness is that the number of offspring generated by individuals with certain properties should be balanced. Different mechanisms for spreading the individuals in the population over the Pareto front have been proposed. In NSGA-II [1] a uniform spread over the Pareto front should be achieved by using a crowded comparison operator that gives individuals in less crowded regions a higher priority. SPEA2 [10] uses a density estimator such that the fitness of an individual is given by its objective vector and a density value which depends on the other individuals in the population. The goal of the density estimator is also to give individuals in less crowded regions a higher priority. Our aim is to get a theoretical understanding how such fairness mechanisms influence the optimization process.

The theoretical understanding of the runtime behavior of MOEAs is far behind their practical success. The first rigorous runtime analyses of such algorithms have been carried out by Laumanns et al. [7] on some pseudo-Boolean functions. They have investigated a mutation-based MOEA called Simple Evolutionary Multi-objective Optimizer (SEMO) that searches locally by flipping in each mutation step a single bit.

* The second author was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

In addition, they have considered a MOEA called Fair Evolutionary Multi-objective Optimizer (FEMO) and shown that this algorithm outperforms SEMO on a particular pseudo-Boolean function called *LOTZ* (*Leading Ones, Trailing Zeroes*). Giel [5] has investigated SEMO with a mutation operator that searches globally and called the algorithm Global SEMO. Global SEMO has also been considered for some well-known combinatorial optimization problems [3,8,9].

In this paper, we want to put forward the runtime analysis of MOEAs and consider how the use of fairness can influence the runtime behavior. We investigate the concept of fairness introduced by Laumanns et al. [7]. The implementation of this concept relies on several counters, where each individual in the population corresponds to one of these counters. The counters measure the number of offspring that the corresponding group of individuals has created. Fairness means to balance these counters to achieve that all groups have been granted the same chance to create a better individual. There are two basic ideas to link individuals with counters. The first idea is that individuals with the same decision vector share a counter and the second idea is that individuals with the same objective vector share a counter. Our goal is to compare the runtime behavior of these two variants.

The outline of this paper is as follows. A short introduction into multi-objective optimization and the algorithms that are subject of our analyses are presented in Section 2. The differences between the two variants of fairness are worked out in Sections 3 and 4. Finally, we finish with some concluding remarks.

2 Algorithms

We start with some basic notations and definitions that will be used throughout the paper. We denote the set of all Boolean values by $\mathbb{B}$ and the set of all real numbers by $\mathbb{R}$ and investigate the maximization of functions $f\colon \mathbb{B}^n \to \mathbb{R}^m$. We call f *objective function*, $\mathbb{B}^n$ *decision space*, and $\mathbb{R}^m$ *objective space*. The elements of $\mathbb{B}^n$ are called *decision vectors* and the elements of $\mathbb{R}^m$ *objective vectors*. We define that y *weakly dominates* y', denoted by $y \succeq y'$, if and only if $y_i \geq y'_i$ for all $i \in \{1, \dots, m\}$, and y *dominates* y', denoted by $y \succ y'$, if and only if $y \succeq y'$ and $y \neq y'$, where $y = (y_1, \dots, y_m) \in \mathbb{R}^m$ and $y' = (y'_1, \dots, y'_m) \in \mathbb{R}^m$ are two objective vectors.

The set $\mathcal{F}_f := \{y \in f(\mathbb{B}^n) \mid \nexists y' \in f(\mathbb{B}^n)\colon y' \succ y\}$ is called the *Pareto front of* f and the set $\mathcal{P}_f := f^{-1}(\mathcal{F}_f) = \{x \in \mathbb{B}^n \mid \nexists x' \in \mathbb{B}^n\colon f(x') \succ f(x)\}$ the *Pareto set of* f. The elements of $\mathcal{F}_f$ and $\mathcal{P}_f$ are called *Pareto optimal*. The set $\{(x, f(x)) \mid x \in \mathcal{P}_f\}$ constitutes the canonical solution of an optimization problem of the considered kind. In the literature a set $\{(x, f(x)) \mid x \in X\}$ with $X \subseteq \mathcal{P}_f$ is also considered as a valid solution if $f(X) = \mathcal{F}_f$. This means that it is sufficient to determine for all Pareto optimal objective vectors $y \in \mathcal{F}_f$ at least one decision vector $x \in \mathbb{B}^n$ with $f(x) = y$.

Laumanns et al. [7] argue that it can be beneficial when all individuals in the population have created roughly the same number of offspring and therefore introduced the algorithm FEMO. This algorithm works with a local mutation operator and uses a counter for each individual in the population to measure the number of offspring the corresponding individual has created. We investigate generalized variants of FEMO. Our algorithms apply a global mutation operator and additionally accept individuals with the same objective vector as an individual in the population. The use of a global mutation operator seems more appropriate as the ability to flip two or more bits in a single mutation step is essential to escape from a local optimum. The relaxed acceptance rule also tends to improve the optimization, since it allows the exploration of plateaus, i. e., regions in the decision space whose decision vectors are mapped to the same objective vector. We distinguish two kinds of fairness depending on whether the fairness is ensured in the decision or objective space. The following algorithm uses fairness in the decision space.

Algorithm 1. *Global FEMO$_{ds}$*

```
1. Choose $x \in \mathbb{B}^n$ uniformly at random.
2. Set $c(x) := 0$.
3. Set $P := \{x\}$.
4. Repeat
     – Choose $x \in \{y \in P \mid c(z) \geq c(y)$ for all $z \in P\}$ uniformly at random.
     – Set $c(x) := c(x) + 1$.
     – Create an offspring $x'$ by flipping each bit of $x$ with probability $1/n$.
     – If there is no $y \in P$ with $f(y) \succ f(x')$ then
         • If $x' \notin P$ then set $c(x') := 0$.
         • Set $P := (P \setminus \{y \in P \mid f(x') \succeq f(y)\}) \cup \{x'\}$.
```

Note, that resetting a counter to 0 depends on the individuals in the current population. This implies that the algorithm forgets about counter values for decision vectors that have been seen during the optimization process but are not part of the current population. This phenomenon is of relevance if a decision vector re-enters the population which has been replaced in the meantime by another decision vector which is mapped to the same objective vector. However, we think that this is a natural way of implementing this idea of fairness as EAs are usually limited to the knowledge of the individuals that are contained in the current population. Note, that Global FEMO$_{ds}$ coincides with Global SEMO [3,9], when the counter values do not influence the search process, i. e., $c(x) = 0$ holds for each search point at each time step.

The goal in multi-objective optimization is to find the Pareto front. Thus the question arises whether it might be more beneficial to associate each counter with an objective vector rather than a decision vector, since the latter approach emphasizes the exploration of the objective space. The following algorithm implements fairness in the objective space.

Algorithm 2. *Global FEMO$_{os}$*

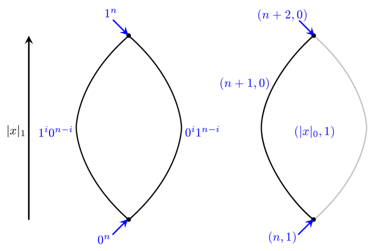
```
1. Choose $x \in \mathbb{B}^n$ uniformly at random.
2. Set $c(f(x)) := 0$.
3. Set $P := \{x\}$.
4. Repeat
     – Choose $x \in \{y \in P \mid c(f(z)) \geq c(f(y))$ for all $z \in P\}$ uniformly at random.
     – Set $c(f(x)) := c(f(x)) + 1$.
```

- *Create an offspring x' by flipping each bit of x with probability $1/n$.*
- *If there is no $y \in P$ with $f(y) \succ f(x')$ then*
  - *If $f(x') \notin f(P)$ then set $c(f(x')) := 0$.*
  - *Set $P := (P \setminus \{y \in P \mid f(x') \succeq f(y)\}) \cup \{x'\}$.*

For our theoretical investigations carried out in the following sections, we count the number of iterations until a desired goal has been achieved. Since we are interested in the discovery of all Pareto optimal objective vectors, we count the number of iterations until an individual for each objective vector of $\mathcal{F}_f$ has been included into the population and call it the optimization time of the algorithm. The expectation of this value is called the expected optimization time.

3 Advantages of Fairness in the Decision Space

The goal of the next two sections is to point out the differences that the use of different fairness mechanisms might have. Therefore we examine situations where the runtime behavior of the two variants differs significantly. To ease the notation in the following sections we will refer to the number of 0- and 1-bits in a decision vector $x \in \mathbb{B}^n$ as $|x|_0$ and $|x|_1$, respectively. We start with the examination of a situation, where Global FEMO$_{ds}$ is efficient while Global FEMO$_{os}$ is inefficient, and therefore investigate the bi-objective function *PL* (*PLateau*) [4]. The function is similar to the well-known single-objective function *SPC* (*Short Path with Constant values*) [6]. *PL* is illustrated in the right figure and defined as follows:



(a) decision space (b) objective space

$$PL(x) := \begin{cases} (|x|_0, 1) & x \notin \{1^i0^{n-i} \mid 1 \le i \le n\}, \\ (n+1, 0) & x \in \{1^i0^{n-i} \mid 1 \le i < n\}, \\ (n+2, 0) & x = 1^n. \end{cases}$$

The function features the following properties. The decision space is partitioned into a short path $SP := \{1^i0^{n-i} \mid 1 \le i \le n\}$ and its complement $\mathbb{B}^n \setminus SP$. The second objective of the function ensures that decision vectors from one of the mentioned sets are comparable while decision vectors from different sets are incomparable. The Pareto front of *PL* is $\mathcal{F}_{PL} = \{(n, 1), (n+2, 0)\}$ and the Pareto set of *PL* is $\mathcal{P}_{PL} = \{0^n, 1^n\}$. The set $SP \setminus \{1^n\}$ constitutes a plateau, since all decision vectors are mapped to the objective vector $(n + 1, 0)$, while $\mathbb{B}^n \setminus SP$ features a richer structure. Since $PL(x) \succ PL(x')$ for $x, x' \in \mathbb{B}^n \setminus SP$ iff $|x|_0 > |x'|_0$, the algorithms are directed to the Pareto optimal decision vector 0^n. This function has already been considered by Friedrich et al. [4] who have shown that Global SEMO is inefficient on *PL*. The next theorem shows that Global FEMO$_{os}$ is also not efficient on this function.

Theorem 1. *The optimization time of Global FEMO$_{os}$ on PL is lower bounded by $2^{\Omega(n^{1/4})}$ with probability $1 - 2^{-\Omega(n^{1/3})}$.*

Proof. We show that the decision vector 1^n is not created with probability $1-2^{-\Omega(n^{1/3})}$ within a phase of $2^{\Omega(n^{1/4})}$ steps. The initial individual $x \in \mathbb{B}^n$ does not belong to *SP* with probability $1 - |SP|/2^n = 1 - 2^{-\Omega(n)}$, as it is chosen uniformly at random. In addition, $|x|_1 \leq 2n/3$ holds with probability $1 - 2^{-\Omega(n)}$ using Chernoff bounds. In the remainder of the proof we consider a typical run consisting of phases of length $n^{3/2}$.

Claim. Within the first $n^{3/2}$ steps with probability $1 - 2^{-\Omega(n^{1/3})}$, the population P never contains 1^n and at one time the population $P = \{0^n, 10^{n-1}\}$ is reached.

Proof. The probability that a mutation flips at least i bits is upper bounded by

$$\binom{n}{i} \cdot \left(\frac{1}{n}\right)^i \leq \left(\frac{en}{i}\right)^i \cdot \left(\frac{1}{n}\right)^i = \left(\frac{e}{i}\right)^i.$$

Therefore the probability that a mutation flips at least $n^{1/3}$ bits is upper bounded by $(e/n^{1/3})^{n^{1/3}} = 2^{-\Omega(n^{1/3}\log n)}$. This implies that none of the first $n^{3/2}$ mutations flips more than $n^{1/3}$ bits with probability $1 - 2^{-\Omega(n^{1/3}\log n)}$.

The probability to create and accept an offspring x' with more 1-bits than its parent is at most $1/n$, since x is required to be in *SP*. Hence, the expected number of such steps is upper bounded by $n^{1/2}$. Due to Chernoff bounds this happens at most $2n^{1/2}$ times with probability $1 - 2^{-\Omega(n^{1/2})}$. Hence, the number of 1-bits increases by at most $2n^{1/2} \cdot n^{1/3} = o(n)$ which implies that the decision vector 1^n has not been found.

As at most $\frac{1}{2} \cdot n^{3/2}$ mutation trials are allocated to $c((n+1, 0))$, the individuals from $\mathbb{B}^n \setminus SP$ are chosen at least $\frac{1}{2} \cdot n^{3/2}$ times for mutation. We consider the first $\frac{1}{4} \cdot n^{3/2}$ of these mutation steps and show that the search point 0^n is included into the population. The probability that an offspring x' of an individual $x \in \mathbb{B}^n \setminus SP$ contains less 1-bits than x and does not belong to *SP* is lower bounded by $(|x|_1 - 1)/en$ if $|x|_1 \geq 2$ and $1/en$ if $|x|_1 = 1$. Therefore the decision vector 0^n is found after an expected number of

$$en + \sum_{i=2}^{n-1} \frac{en}{i-1} \leq en + en(\ln(n-2)+1) \leq en(\ln n + 2)$$

individuals from $\mathbb{B}^n \setminus SP$ have been chosen for mutation. Using Markov's inequality the probability to discover the decision vector 0^n within $2en(\ln n + 2)$ steps is at least $1/2$. Dividing $\frac{1}{4} \cdot n^{3/2}$ steps into $n^{3/2}/(8en(\ln n + 2)) = \Omega(n^{1/3})$ phases of length $2en(\ln n + 2)$ the decision vector 0^n is reached with probability at least $1 - 2^{-\Omega(n^{1/3})}$. The remaining $\frac{1}{4} \cdot n^{3/2}$ of these mutation steps affect 0^n. Therefore the search point 10^{n-1} is included into the population with probability $1 - 2^{-\Omega(n^{1/2})}$ using similar arguments. □

We now consider an additional phase of length $n^{3/2}$. Within this phase a search point with more than $n/2$ 1-bits is not included into the population using previous arguments. Additionally, a situation is reached where $c(n, 1) = c(n+1, 0)$ holds. From this point of time the two individuals with objective vectors $(n, 1)$ and $(n+1, 0)$ are alternately selected for mutation. We consider the situation when $c(n, 1) = c(n+1, 0)$ for the first time and show the following invariant to complete the proof.

Claim. Assume that $0^n \in P$ and $\max_{x \in P} |x|_1 \leq (n/2)$. Consider a non-empty phase of at most $n^{3/2}$ steps. Then with probability $1-2^{-\Omega(n^{1/3})}$, the population never contains 1^n and at one time a population P with $0^n \in P$ and $\max_{x \in P} |x|_1 \leq (n/2)$ is reached.

Proof. The search point 0^n will not be removed from the population once it has been included. From the proof of the previous claim, we already known that the decision vector 1^n is not obtained within a phase of $n^{3/2}$ steps with probability $1 - 2^{-\Omega(n^{1/3})}$. The decision vector 0^n is selected at least $\frac{1}{2} \cdot n^{3/2} - 1$ times for mutation within the considered phase. With probability at least $1/(en)$ such a mutation produces the search point 10^{n-1}. Hence, within the considered phase of length $n^{3/2}$ this holds with probability $1 - 2^{-\Omega(n^{1/3})}$. Having produced the search point 10^{n-1}, it replaces the previous search point of *SP* in the population. Hence, the assumption of the claim is fulfilled again. □

Considering the invariant at most $2^{n^{1/4}}$ times, Global FEMO$_{os}$ does not create the decision vector 1^n with probability $1 - 2^{-\Omega(n^{1/3})}$. This proves Theorem 1 as all failure probabilities are bounded by $1 - 2^{-\Omega(n^{1/3})}$. □

We will see that Global FEMO$_{ds}$ performs much better on *PL* than its counterpart Global FEMO$_{os}$. The main reason for this is that after a while the Pareto optimal decision vector 0^n is prevented from generating additional offspring that can stop the random walk on the plateau.

Theorem 2. *The expected optimization time of Global FEMO$_{ds}$ on PL is $\mathcal{O}(n^3 \log n)$.*

Proof. Before showing that Global FEMO$_{ds}$ quickly creates the decision vectors 0^n and 1^n we summarize some results concerning *PL*. On one hand, the decision vector 0^n is created with probability at least $1/2$ if at least $\gamma n \log n$ individuals not from *SP* are chosen for mutation, where $\gamma > 0$ is a constant (see proof of Theorem 1). On the other hand, the decision vector 1^n is created with probability at least $1/2$ if at least δn^3 individuals from *SP* are chosen for mutation and all offspring of individuals not contained in *SP* do not belong to *SP*, where $\delta > 0$ is an appropriate constant (see [6]).

We show that the expected time until one decision vectors of $\{0^n, 1^n\}$ is introduced into the population is $\mathcal{O}(n^3 \log n)$. We observe a phase of length

$$\ell := (2\gamma \log n + 1) \cdot (\delta n^3 + \gamma n \log n) = \mathcal{O}(n^3 \log n)$$

and distinguish two cases. If at least $\gamma n \log n$ individuals not from *SP* are chosen for mutation, the probability to find the decision vector 0^n is lower bounded by $1/2$ according to the first statement. The probability that an offspring of an individual not from *SP* belongs to *SP* is upper bounded by $1/n$. Therefore otherwise at most $2\gamma \log n$ offspring of individuals not from *SP* belong to *SP* with probability at least $1/2$ according to Markov's inequality. Assuming that this has happened and applying the pigeonhole principle we can be sure that the phase contains a sub-phase of length

$$\delta n^3 + \gamma n \log n,$$

where no offspring of individuals not contained in *SP* belong to *SP*. The mentioned sub-phase fulfills the second statement, since at least δn^3 individuals from *SP* are selected for mutation. Hence, the decision vector 1^n is created with probability at least $1/4$. Since the probability to create the decision vector 0^n or 1^n in a phase of length ℓ is lower bounded by $1/4$, an expected number of at most $4\ell = \mathcal{O}(n^3 \log n)$ steps suffices.

We now consider the situation where the decision vector 0^n has been found and the decision vector 1^n is still missing. Observe a phase of length

$$\ell' := (2e\ln(2\delta n^3) + 1) \cdot (\delta n^3 + en\ln(2\delta n^3)) = \mathcal{O}(n^3 \log n).$$

If 0^n is selected at most $en\ln(2\delta n^3)$ times then the probability that at most $2e\ln(2\delta n^3)$ offspring of 0^n are from *SP* is lower bounded by $1/2$ using Markov's inequality. Assuming that this has happened the phase contains a sub-phase of length

$$\delta n^3 + en\ln(2\delta n^3)$$

in which at least δn^3 individuals from *SP* are chosen for mutation and all offspring of the individual 0^n do not belong to *SP*. Hence, the probability that the missing decision vector 1^n is found or the counter value $c(0^n)$ exceeds $en\ln(2\delta n^3)$ is lower bounded by $1/4$. One of the mentioned events occurs after an expected number of most $4\ell' = \mathcal{O}(n^3 \log n)$ steps. If the individual 1^n still has not been found we observe a phase of length $2en^2 + \delta n^3$. The probability to add a new individual from *SP* to the population is lower bounded by $1/(en^2)$ as at most 2 specific bits have to flip. This worst case occurs if 0^n is selected for mutation and 10^{n-1} is already contained in the population. Hence, the probability that in the first $2en^2$ steps of the phase a new individual from *SP* with an initial counter value of 0 is added to the population is lower bounded by $1/2$ due to Markov's inequality. Assuming that this has happened the probability that the individual 0^n is selected in the following δn^3 steps can be upper bounded as follows. The probability to reset the counter of the individual from *SP* is lower bounded by $1/en$. The probability that this does not happen in $en\ln(2\delta n^3)$ consecutive steps is upper bounded by

$$\left(1 - \frac{1}{en}\right)^{en\ln(2\delta n^3)} \leq e^{-\ln(2\delta n^3)} = \frac{1}{2\delta n^3}.$$

The probability that this does not happen in a phase of length δn^3 is upper bounded by $\delta n^3 \cdot 1/(2\delta n^3) \leq 1/2$. We conclude that the counter value of the actual individual from *SP* does not exceed $en\ln(2\delta n^3)$ with probability at least $1/2$ and therefore the individual 0^n is not chosen for mutation. Assuming that this has happened the probability that the decision vector 1^n is found is lower bounded by $1/2$. Hence, the decision vector 1^n is found in an expected number of $8 \cdot (2en^2 + \delta n^3) = \mathcal{O}(n^3)$ steps.

We also have to examine the situation that the decision vector 1^n has been found and the decision vector 0^n is still missing. We wait until the population contains an additional individual not contained in *SP* and the counter value $c(1^n)$ is at least as big as the counter value of this individual. Afterwards we observe a phase of length $2\gamma n \log n$. We can be sure that at least $\gamma n \log n$ steps are allocated to individuals not from *SP* as $c(1^n)$ is never set to 0. Hence, after an expected number of $\mathcal{O}(n \log n)$ additional steps the decision vector 0^n is added to the population. □

4 Advantages of Fairness in the Objective Space

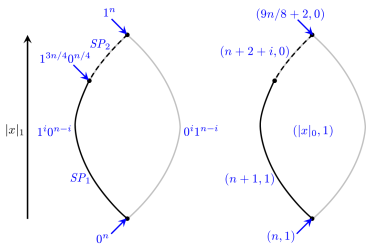
In this section, we point out situations where the use of fairness in the objective space favors over fairness in the decision space. We have already seen that the latter fairness mechanism enables a random walk on a plateau of constant fitness where the former fairness mechanism does not allow this kind of exploration. During the random walk the counter of the individual on the plateau is set to 0 whenever a new individual on the plateau is created. This can also be a drawback of fairness in the decision space as it might prevent the algorithm from improvements that are harder to obtain than finding a new individual on the plateau.

The function that is used to point out the mentioned behavior is similar to the function *PL* that has been examined in Section 3. To ease the following definition we assume $n = 8m$, $m \in \mathbb{N}$, and define

$$SP_1 := \{1^i 0^{n-i} \mid 1 \leq i \leq 3n/4 - 1\}$$

and

$$SP_2 := \{1^{3n/4+2i} 0^{n/4-2i} \mid 0 \leq i \leq n/8\}.$$



The function *PLG* (*PLateau and Gaps*) is illustrated in the figure to the right and defined as follows:

$$PLG(x) := \begin{cases} (|x|_0, 1) & x \notin SP_1 \cup SP_2, \\ (n+1, 1) & x \in SP_1, \\ (n+2+i, 0) & x = 1^{3n/4+2i} 0^{n/4-2i}. \end{cases}$$

Note, that $\mathcal{F}_{PLG} = \{(n+1,1), (9n/8+2,0)\}$ and $\mathcal{P}_{PLG} = SP_1 \cup \{1^n\}$. The short path *SP* is divided into a plateau and a short path with little gaps that leads to the second Pareto optimal objective vector $(9n/8+2, 0)$.

The next theorem shows that Global FEMO$_{os}$ performs well on *PLG*.

Theorem 3. *The expected optimization time of Global FEMO$_{os}$ on PLG is $\mathcal{O}(n^3)$.*

Proof. An individual of $SP_1 \cup SP_2$ is added to the population after an expected number of $\mathcal{O}(n \log n)$ steps, since before the achievement of such a situation the population contains one individual and the algorithm behaves like (1+1) EA on ONEMAX (see [2]).

We first consider the situation where this individual belongs to SP_1. After an expected number of $\mathcal{O}(n^3)$ steps an individual of SP_2 is introduced into the population (see [6]). The probability to find a better individual of SP_2 under the condition that the individual of SP_2 has been selected for mutation is lower bounded by $(1/n)^2(1 - 1/n)^{n-2} \geq 1/(en^2)$ as it suffices to flip its two leftmost 0-bits. Hence, in expectation at most en^2 attempts per non-optimal individual of SP_2 are needed to improve it. The counter of the Pareto optimal individual of SP_1 is never reset. Hence, the individual of SP_2 is chosen at least once in two consecutive iterations. Therefore, an expected number of at most $2 \cdot n/8 \cdot en^2 = \mathcal{O}(n^3)$ steps is needed to obtain the missing decision vector 1^n.

In the case that the first individual of $SP_1 \cup SP_2$ belongs to SP_2 an individual of $\mathbb{B}^n \setminus SP_2$ is created with probability at least $1/e$ in a mutation step as it suffices to flip a single bit. Hence, after an expected number of $e = O(1)$ steps the population contains besides a solution of SP_2 an additional solution of $\mathbb{B}^n \setminus SP_2$. A decision vector of SP_1 is reached by allocating an expected number of $O(n \log n)$ mutation trials to the individuals of $\mathbb{B}^n \setminus SP_2$. We already know that $O(n^3)$ mutation trials allocated to the individuals of SP_2 are enough to reach the decision vector 1^n which completes the proof. □

The next theorem states that Global FEMO*ds* is inefficient on *PLG*. We will see that the random walk on the plateau prevents the algorithm from following the short path to the second Pareto optimal decision vector 1^n.

Theorem 4. *The optimization time of Global FEMO*ds* on PLG is lower bounded by* $2^{\Omega(n^{1/2})}$ *with probability* $1 - 2^{-\Omega(n^{1/2})}$.

Proof. For the initial individual x holds $|x|_1 > 5n/8$ with probability $e^{-\Omega(n)}$ due to Chernoff bounds. One of the first $2^{n^{1/2}}$ mutations flips more than $n^{1/2}$ bits with probability $2^{-\Omega(n^{1/2}\log n)}$ (cf. proof of Theorem 1). We assume that these events have not happened and show that 1^n is not found within a phase of length $2^{n^{1/2}}$ w. h. p.

We wait until the algorithm has generated for the first time an individual $x \in SP_2$ with $|x|_1 \geq 3n/4 + n^{1/2} - 1$. As at most $n^{1/2}$ bits flip per mutation, we can be sure that $|x|_1 \leq 3n/4 + 2n^{1/2} - 2$ holds in the next step and that the population contains an additional individual of SP_1. The probability to generate a better individual of SP_2 under the condition that the individual of SP_2 has been selected for mutation is upper bounded by $1/n^2$ since at least the two leftmost 0-bits of x have to be flipped. The probability that $n^2 - 1$ trials to find a better individual of SP_2 fail is lower bounded by $(1 - 1/n^2)^{n^2-1} \geq 1/e$. As at most $n^{1/2}$ bits flip per mutation, the algorithm is at least

$$\frac{n/4 - 2n^{1/2} + 2}{n^{1/2}} = \frac{n^{1/2}}{4} - 2 + \frac{2}{n^{1/2}} \geq \frac{n^{1/2}}{8}$$

times in the above situation. Hence, the probability that there is an individual $x^* \in SP_2$ for which the first $n^2 - 1$ trials to find a better individual of SP_2 fail is at least

$$1 - \left(1 - \frac{1}{e}\right)^{n^{1/2}/8} \geq 1 - 2^{-\Omega(n^{1/2})}.$$

We upper bound the counter value of the individual of SP_1 which shows that the algorithm is not able to find an individual with more 1-bits than x^*. Note, that there is at least one Hamming neighbor for the individual of SP_1 that is mapped to the same objective vector. Hence, the probability to reset the counter value of the individual of $P \cap SP_1$ is lower bounded by $1/en$. Therefore, the probability that the counter value of an individual of SP_1 reaches n^2 is upper bounded by

$$\left(1 - \frac{1}{en}\right)^{n^2-1} = \left(1 - \frac{1}{en}\right)^{en \cdot n/e} \cdot \frac{en}{en-1} \leq e^{-n/e} \cdot \frac{en}{en-1} = 2^{-\Omega(n)}.$$

As the probability that this happens in the observed phase is upper bounded by $2^{n^{1/2}} \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$, the statement of the theorem follows. □

5 Conclusions

Popular variants of MOEAs such as NSGA-II or SPEA2 use specific modules to explore the Pareto front of a given problem by favoring solutions belonging to regions in the decision or objective space that are rarely covered. With this paper, we have taken a first step to understand such mechanisms by rigorous runtime analyses. We have shown that there are simple plateau functions which cannot be optimized without fairness or with fairness in the objective space, but with a MOEA which implements fairness in the decision space (cf. Section 3). We also proved that for certain "perforated" plateaus the impact of fairness can be the other way around (cf. Section 4).

References

1. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN VI 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
2. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81 (2002)
3. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. In: Proceedings of Conference on Genetic and Evolutionary Computation (GECCO), vol. 1, pp. 797–804. ACM Press, New York (2007)
4. Friedrich, T., Hebbinghaus, N., Neumann, F.: Plateaus can be harder in multi-objective optimization. In: Proceedings of Congress on Evolutionary Computation (CEC), pp. 2622–2629. IEEE Press, Los Alamitos (2007)
5. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proceedings of Congress on Evolutionary Computation (CEC), pp. 1918–1925. IEEE Press, Los Alamitos (2003)
6. Jansen, T., Wegener, I.: Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Transactions on Evolutionary Computation 5(6), 589–599 (2001)
7. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. IEEE Transactions on Evolutionary Computation 8(2), 170–182 (2004)
8. Neumann, F.: Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. European Journal of Operational Research 181(3), 1620–1629 (2007)
9. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Natural Computing 5(3), 305–319 (2006)
10. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Proc. of EUROGEN 2001, pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE) (2002)

The Parallel Predator-Prey Model: A Step towards Practical Application

Christian Grimme, Joachim Lepping, and Alexander Papaspyrou

Robotics Research Institute - Section Information Technology,
Dortmund University of Technology, 44221 Dortmund, Germany
{christian.grimme,joachim.lepping,alexander.papaspyrou}@udo.edu

Abstract. In this paper, we apply the parallel predator-prey model for multi-objective optimization to a combinatorial problem for the first time: Exemplarily, we optimize sequences of 50 jobs for an instance of the bi-criteria scheduling problem $1|d_j|\sum C_j, L_{max}$ with this approach. The modular building block architecture of the predator-prey system and the distribution of acting entities enables the analysis of separated problem knowledge and the design of corresponding variation operators. The actual modules are derived from local heuristics that tackle fractions of the complete problem. We unveil that it is possible to cover different areas of the Pareto-front with special property operators and make evident that the whole front can be covered if those operators are applied simultaneously to the spatial population. Further, we identify open problems that arise when the predator-prey model is applied to combinatorial problems which have not yet occurred for real-valued optimization problems.

1 Introduction

Ten years ago, Laumanns et al. [6] proposed a parallel evolutionary algorithm to tackle multi-objective optimization (MOO) problems. Following the predation paradigm from biology, he adapted the—partially simplified—interaction relations between predators and preys: a population of prey is distributed on a spatial structure that is represented by an undirected graph. Predators move randomly along the edges in order to chase those preys weak against their certain criterion. The presence of several predators—each representing a single criterion—was expected to force the prey to likewise adapt to the threats from all predators and thus result in suitable trade-off solutions for MOO problems.

During the following years, Deb [1], Li [7], and Schmitt et al. [8] adopted this approach, but modified critical points of the algorithm. Deb re-aggregated the disjoint selection mechanism, implicitly losing effect of independent working predator agents. Li introduced stronger parallelism by allowing the prey to move, however at the expense of simplicity and thus perceivability. Schmitt et al. removed parts of the implicit parallelism from the system by controlling the prey reproduction in a centralized manner to establish a $(\mu \overset{+}{,} \lambda)$-selection in larger neighborhoods to foster convergence at the cost of deteriorated diversity.

Recently, Grimme and Lepping [3] reclaimed the original algorithm and proposed a methodology to utilize both the parallel character and the distributed influence of predators, enabling an easy parameterization for special problem instances with detail knowledge on particular objectives. The main idea of this approach was to consider the reproduction mechanism, variation operators, predator movement, and other characteristics of predators as building blocks. This allows the creation of predator species on the basis of certain characteristics such as variation operators and movement patterns to the point of specific modules tailored to tackle a certain aspect of the inspected multi-objective problem. The parallel impact of those predator species eventually delivers compromise solutions to the overall MOO problem.

Up to now all evaluations of the model where founded on very simple real-valued problem instances that where useful to understand the model dynamics. However, the large class of problems from combinatorial optimization, that represents more practical or even real-world problems, has not been analyzed so far. In this paper, we perform a first investigation of such problems and try to find out whether the proposed building block approach for variation operators is also applicable for combinatorial multi-objective optimization. As problem domain we chose scheduling because of its frequent occurrence and indisputable relevance to many areas such as production, logistics, and information technology. Additionally, most scheduling problems belong to the class of $\mathcal{NP}$-complete problems. This is all the more true for multi-objective scheduling problem which makes it frequently impossible to efficiently find optimal solutions. However, we chose as a first step the only bi-criteria scheduling problem for which (at least to the authors' knowledge) the Pareto-front can be computed within polynomial time. We apply the predator-prey algorithm on the basis of well-known heuristics for each single objective and show that—using this approach—the optimal Pareto-front can be reached.

The rest of the paper is organized as follows: in Section 2, we describe the original problem and discuss solution strategies for each single objective. Then, in Section 3, we shape the building blocks of our predator-prey environment. Next, in Section 4, we evaluate the performance of the afore defined environment on the problem. Finally, in Section 5, we conclude our work, giving an outlook on future investigations.

2 A Multi-objective Scheduling Problem

For the multi-objective scheduling problem we assume a single machine as the simplest of all machine environments with n jobs having to be processed on it. The combinatorial optimization task is to find all Pareto-optimal sequences of job executions concerning the two objectives. Once such a sequence of jobs has been determined those jobs are dispatched on the machine and executed without any forced delay. Consequently, the resulting schedule can be classified as a *non-delay* schedule.

A job $j \in J$ is given by two simple properties which characterize the problem instance: the processing time p_j, and the due date d_j of job j. The latter should be greater or equal than the processing time ($d_j \geq p_j$), representing the committed shipping or completion date promised to the customer. The time that job j exits on the machine is denoted by C_j and naturally depends on the scheduling sequence. If the completion time C_j is after the due date d_j a penalty is incurred and the job is denoted late.

The plethora of multi-objective scheduling problems includes only very few that can be solved with polynomial time complexity. In order to allow for benchmarking the obtained solutions' quality in terms of convergence and diversity, a problem that has algorithms for finding optimal solutions in reasonable time would be favorable.

As objective we assume the on the one hand *maximum lateness* which is given by $L_{max} = \max_{j=1\ldots n}\{L_j\}$ with $L_j = C_j - d_j$ being the lateness of job j. This objective measures the worst violation of due dates. On the other hand we consider the *total completion time* of all n jobs, computed by $\sum_{j=1}^{n} C_j$, which expresses the concern of finishing each job as early as possible. While $\sum C_j$ objective completely ignores the due date L_{max} objective refers to due dates only and ignores the jobs' processing times. This leads to the conflicting nature of the objectives.

Further, we use for the description of scheduling problems the $\alpha|\beta|\gamma$ notation of Graham [2], where α denotes the machine environment, β the constraints, and γ the objective field respectively. According to this convention the multi-objective scheduling problem can be formulated as $1|d_j|\sum C_j, L_{max}$.

For this problem an efficient algorithm for computing an optimal solution already exists: In 1980 van Wassenhoven and Gelder's [11] presented a polynomial-time algorithm with an overall complexity of $\mathcal{O}(n^3 \log n)$ that iteratively generates all Pareto-optimal trade off solutions by executing two nested loops. For a detailed description along with examples please refer to T'kindt and Billaut [10].

2.1 Solution Strategies

The objectives for described problem are the maximum lateness and total completion time which are usually conflicting: although there might be job sequences for which the optimal solutions for each of the two objectives do not disrupt each other[1], most problem instances do not have matching orderings and thus produce contradictory solutions for the singe-objective case.

For the single-objective scheduling problem $1|d_j|L_{max}$ which considers only maximum lateness, Jackson presented in 1955 [5] the Earliest Due Date (EDD) rule as an optimal algorithm. It sequences the jobs in non-decreasing order of their due dates. This simple rule gives an optimal sequence for the single machine maximum lateness problem, and is used as a heuristic for numerous other scheduling problems. For total completion time objective and the corresponding single-objective scheduling problem $1||\sum C_j$, Smith developed in 1956 [9] the

[1] That is, when the optimal sequence regarding maximum lateness is identical to the optimal sequence regarding total completion time.

Shortest Processing Time (SPT) rule. This sorting-based dispatch rule sequences all jobs in non-decreasing order of their processing times p_j and optimally solves the given problem. Both extremal solutions can be determined easily in logarithmic time.

3 Instantiation of the Predator-Prey Model

We base our setup on the building-block based predator-prey model defined by Grimme et al. [4]: Predators comprise a specific objective, a set of variation operators for reproduction, a neighborhood function for determining the individuals that are exposed to selection and reproduction, and a walking function for the movement pattern on the spatial structure; Prey represent the solutions of the MOO problem.

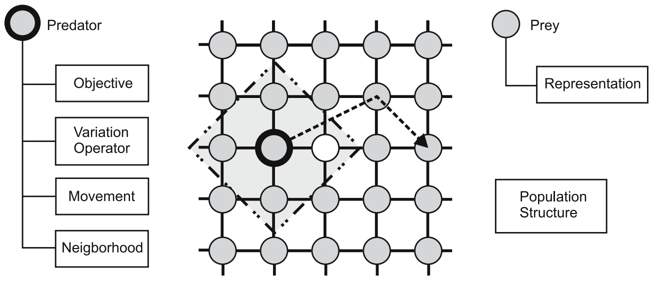


Fig. 1. Schematic depiction of the building-block based predator-prey model. The depicted instance exemplarily shows a toroidal population structure that hosts prey individuals along with a single predator. The displayed movement involves two distinct walks (from positions $2, 3$ to $4, 2$ and from positions $4, 2$ to $5, 3$), each obeying the graph's vertices. Additionally, a neighborhood with a radius of 1 is shown, as well as one free vertex from which the prey has been removed.

A schematic depiction of this concept is shown in Figure 1. Following, we instantiate the model by discussing the problem encoding, introducing the applied operators, and concretizing the remaining building blocks.

3.1 Encoding of the Problem

Since the reviewed scheduling problem bases on a sequence of n jobs, we use a standard permutation encoding to represent the genotype of the problem. As we consider an offline scheduling problem, it is the task of the algorithm to find the set of optimal job permutations.

3.2 Structure of the Applied Variation Operators

Former work has shown that the understanding of how each variation operator influences the population and the subsequent combination of these using autonomously acting predators yielded good approximation results. In the context of a practical problem, the different variation operators can be based on local search heuristics that assess a certain aspect of the considered multi-objective problem, resulting in an accelerated convergence towards the corresponding fraction of the Pareto-front.

For the considered scheduling problem, we apply this very methodology by integrating problem specific knowledge into the operator design: The variation operators used for the reproduction of prey utilize the strategies for single- objective optimal solution described in Section 2.1. The following paragraphs describe these operators in detail.

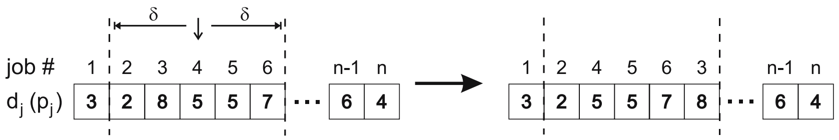


Fig. 2. Schematic depiction of the working principle of the EDD (SPT) mutation operator with $\delta = 2$

SPT mutation is a local mutation heuristic that integrates the SPT solution strategy for the $1||\sum C_j$ problem. As this strategy potentially results in an optimal solution for just one of the considered objectives, the mutation operator is designed to be applied only locally. Figure 2 depicts the application of this operator to a given sequence: a position k is selected randomly in the permutation representation of the genotype. Then, a subsequence of $2\delta + 1$ genes are sorted according to SPT. The size of this δ-neighborhood is determined by a normal distribution with an externally adjustable step size of σ. Obviously, $\delta = 0$ has no effect as only the initial gene at position k is selected. On the other hand, a growing δ leads to a higher probability of a completely SPT-ordered genome which limits the Pareto-front regarding this objective[2].

EDD mutation works completely analogous to the SPT mutation, using the Earliest Due Date sorting strategy. As described in Section 2.1, it delivers optimal solutions for $1|d_j|L_{max}$.

Random Swap (RS) mutation is rather a classic mutation operator than a local search heuristic. For a given number ζ is swaps ζ times a pair of genes in the considered genome. This variation operator is expected to bring innovation to the population and reduce the effect of the heuristic tendency to extremal solutions.

3.3 Defining the Remaining Building Blocks

Finally, we need to instantiate three additional building blocks in order to complete the model instantiation for the given problem.

[2] Since it represents the optimal solution for $\sum C_j$ and thus one extremal point on the Pareto-front.

The *spatial population structure* is represented by a two-dimensional toroidal grid with a size of 40×40, which is initialized with random individuals. The *movement* of a predator follows a random walk pattern, ensuring that each position is visited equally often. The *number of evaluations* for each model run was restricted to 30,000. The *selection neighborhood* of a predator is fixed to a radius of 1, resulting in a selection set of five prey individuals.

4 Evaluation

In order to assess the applicability of the predator-prey approach to our problem, we generated a single instance of $1|d_j|\sum C_j, L_{max}$ with $n = 50$ jobs.

Herefor, the processing time was sampled using a uniform distribution as $p_j = \mathcal{U}(1, 10),\ \forall j = 1 \dots n$. In order to guarantee that all due dates can be met, we determine correspondingly $d_j = p_j + \mathcal{U}(1, 990),\ \forall j = 1 \dots n$.

This ensures that many Pareto-optimal solutions exist as the widely distributed due dates allow for a larger variety of L_{max} values, a property which was verified by the application of the polynomial algorithm, see Section 2.1, which produces a result with 36 Pareto- optimal solutions that form a well distributed front. A detailed description of the used problem instance can be obtained from the authors' web pages[3].

Our procedure for evaluation involved three steps: (1) the calculation of a reference solution using the aforementioned algorithm, (2) the isolated application of each operator in order to identify the specific effects, and (3) the combined application of all operators and analysis of the achieved results.

4.1 Isolated Application of Each Operator

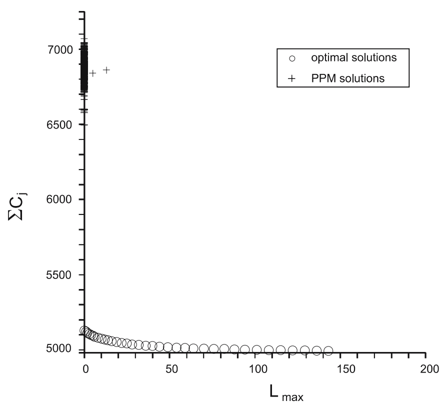
In order to conceptualize a powerful combination of variation operators for the here addressed problem, we first identified the isolated effects of the different single objective operators.

EDD Mutation. Here, we focus on the exclusive influence of EDD mutation and the corresponding single-objective selection. As mutation width, see Section 3.2, we set $\sigma = 5$. Herewith, we made two experiments using a single predator: one L_{max} selection run and one $\sum C_j$ selection run. The outcomes are shown in Figure 3, separated by objectives.

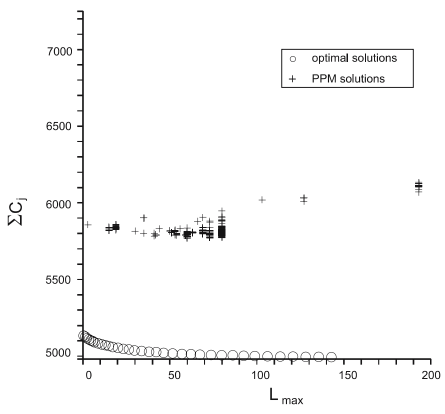
As expected from the theoretical analysis, the EDD mutation solves the problem for the first objective (L_{max}) optimally. Therefore, when the predator selects according to L_{max}, many solutions are found that reach the minimum possible objective value of 0, see Figure 3(a). More interesting is the behavior of the selection regarding total completion time: The solution covers the whole range of the search space while a good diversity is preserved as well. However, no convergence towards the actual front is observable.

As such, the EDD mutation acts—from the total completion time selection's point of view—like a local search at a random point in the genome.

[3] `http://www.it.irf.uni-dortmund.de/~{}lepping/ppsn/`



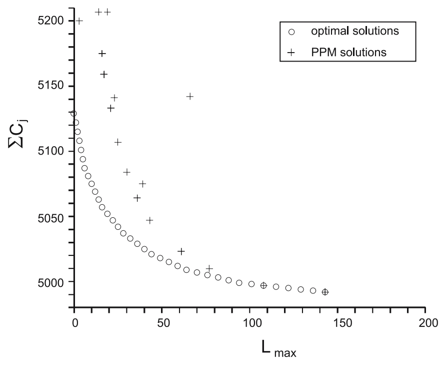
(a) Selection regarding L_{max}



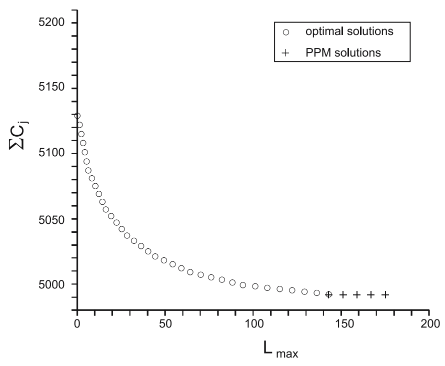
(b) Selection regarding $\sum C_j$

Fig. 3. Evaluation of the EDD mutation operator in combination with two different selections. Additionally, the real Pareto-front is depicted as reference.

SPT Mutation. In analogy to the previous examination, we perform exclusive SPT mutation for both objectives and analyze the corresponding solutions, see Figure 4.



(a) Selection regarding L_{max}



(b) Selection regarding $\sum C_j$

Fig. 4. Evaluation of the SPT mutation operator in combination with two different selections. Additionally, the real Pareto-front is depicted as reference.

It turns out that the SPT mutation finds the optimal $L_{max}(SPT/EDD)$ point on the bottom right edge of the front ($\sum C_j = 4992$). Contrary to EDD mutation, the SPT mutation does not completely fail when applied to the non-related objective, see Figure 4(a): in conjunction with L_{max} selection, it achieves both a good convergence to the front as well as an acceptable diversity. However, the left part of the front, which lies more in the EDD region, cannot be reached with this single operator.

The effect of SPT mutation seems to be stronger than the one of EDD mutation in terms of convergence, see Figure 4(b). Also, it is more robust with respect to the selection objective. The conclusion that can be drawn from this is two-edged: While the SPT mutation is able to favor convergence to the actual front, the population may collapse into one optimal solution when the influence of the operator becomes too strong for the total completion time objective.

RS Mutation. The application of RS mutation reveals a similar effect for both objectives: The order of the jobs is highly disrupted, resulting in a almost complete coverage of the search space far away from the actual front. As such, this operator qualifies for the injection of higher diversity. However, RS mutation must be used carefully as it is counterproductive for the convergence towards the front.

4.2 Combined Application of All Operators

The effects that could be perceived during the isolated application of each of the operators indicate that their combination could lead to a good approximation of the problem's Pareto-front. To this end, we combined the EDD and SPT mutation operators as described in Section 4.1 and additionally introduced and fine-tuned the RS mutation in order to preserve diversity. The final parametrization of each predator is shown in Table 1.

Table 1. Parameterization of the predators for the combined problem solving scenario

Predator	Objective	Mutation	Parameter
P1	L_{max}	SPT	$\sigma = 20$
P2	$\sum C_j$	EDD	$\sigma = 2$
P3	L_{max}	RS	$\zeta = 1$
P4	$\sum C_j$	RS	$\zeta = 10$

The evaluation of the combined run results in the Pareto-front approximation depicted in Figure 5. It turns out that it is possible to combine the beneficial effects to achieve a well overall approximation. However, the intensity of the operators had to be tuned by hand which took several attempts. Nevertheless, it is remarkable that all identified characteristics are preserved in their combined application, resulting in a front that is almost covered as a whole.

5 Conclusion and Future Work

In this work, we applied the concept of multi-objective problem solving through parallel execution of single objective variation operators in the predator-prey model to a bi-criteria scheduling problem. Thus, this is a first step to transfer the hitherto theoretically investigated methodology into the new problem domain of combinatorial optimization. We are able to express problem specific

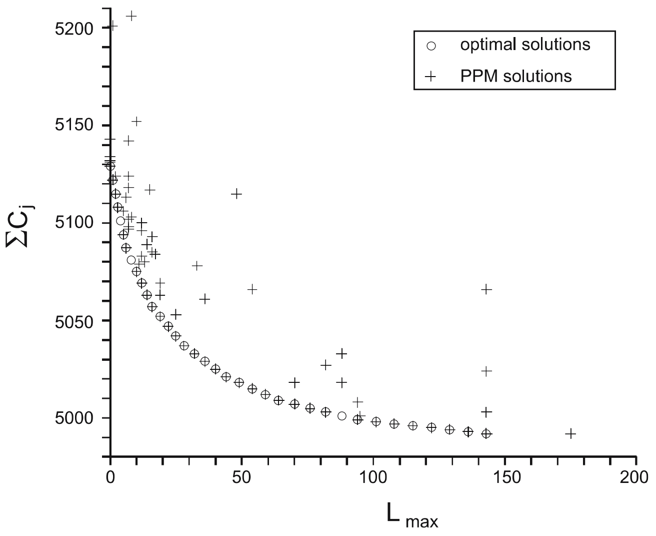


Fig. 5. Parallel application of all mutation operators in combination with the two different selections and the real Pareto-front

single-objective knowledge by corresponding variation operators. Their analysis shows that it is possible to reliably cover certain regions of the Pareto-front. The simultaneous and parallel impact on the population yields a good set of trade-off solutions. Apparently, it is possible to combine observed effects of single operators in order to achieve an exact and diverse solution. This is generally in line with the results that have been obtained for real-valued MOO problems. However, it is much harder to preserve the covered areas in both diversity and convergence when the representing predators are applied simultaneously. Thus, finding a good configuration of the system involves manual tuning that requires sensitivity and experience.

For future work, the process of operator design and model configuration must be simplified and made more reliable. To this end, the model concept must be expanded with capabilities to force trade-off solutions from reachable extremal points by itself. At the moment, the model seems to be constricted by the predators' single-objective selection that prevents the system from finding compromises directly as well as from preserving them. Concepts that focus on niching techniques as building blocks or additional properties of the spacial population structure are promising topics for future investigation.

Acknowledgement

Joachim Lepping is member of the Collaborative Research Center 531, "Computational Intelligence", at Dortmund University of Technology with financial support of the Deutsche Forschungsgemeinschaft (DFG).

References

1. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley-Interscience Series in Systems and Optimization. Wiley, Chichester (2001)
2. Graham, R.L., Lawer, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Annals of Discrete Mathematics 5, 287–326 (1979)
3. Grimme, C., Lepping, J.: Designing Multi-Objective Variation Operators Using a Predator-Prey Approach. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 21–35. Springer, Heidelberg (2007)
4. Grimme, C., Lepping, J., Papaspyrou, A.: Exploring the Behavior of Building Blocks for Multi-Objective Variation Operator Design using Predator-Prey Dynamics. In: Thierens, D., et al. (eds.) Prococeedings of the Genetic and Evolutionary Computation Conference (GECCO), London, United Kingdom, vol. 1, pp. 805–812. ACM Press, New York (2007)
5. Jackson, J.R.: Scheduling a Production Line to Minimze Maximum Tardiness. Management Science Research Project, Research Report 43, University of California, Los Angeles (1955)
6. Laumanns, M., Rudolph, G., Schwefel, H.-P.: A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study. In: Eiben, A.E., Bäck, Th., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 241–249. Springer, Heidelberg (1998)
7. Li, X.: A Real-Coded Predator-Prey Genetic Algorithm for Multiobjective Optimization.. In: Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), pp. 207–221 (2003)
8. Schmitt, K., Mehnen, J., Michelitsch, T.: Using Predators and Preys in Evolution Strategies. In: Beyer, H.-G., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), vol. 1, pp. 827–828. ACM Press, New York (2005)
9. Smith, W.E.: Various Optimizers for Single-stage Production. Naval Research Logistics Quarterly 3, 59–66 (1956)
10. T'kindt, V., Billaut, J.-C.: Multicriteria Scheduling. Theory, Models and Algorithms, 2nd edn. Springer, Berlin (2006)
11. van Wassenhove, L.N., Gelders, F.: Solving a Bicriterion Scheduling Problem. European Journal of Operational Research 2(4), 281–290 (1980)

Functional-Specialization Multi-Objective Real-Coded Genetic Algorithm: FS-MOGA

Naoki Hamada, Jun Sakuma, Shigenobu Kobayashi, and Isao Ono

Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, Nagatsuta. Yokohama 226-8502, Japan
hamada@ic.dis.titech.ac.jp

Abstract. This paper presents a Genetic Algorithm (GA) for multi-objective function optimization. In multi-objective function optimization, we believe that GA should adaptively switch search strategies in the early stage and the last stage for effective search. Non-biased sampling and family-wise alternation are suitable to overcome local Pareto optima in the early stage of search, and extrapolation-directed sampling and population-wise alternation are effective to cover the Pareto front in the last stage. These situation-dependent requests make it difficult to keep good performance through the whole search process by repeating a single strategy. We propose a new GA that switches two search strategies, each of which is specialized for global and local search, respectively. This is done by utilizing the ratio of non-dominated solutions in the population. We examine the effectiveness of the proposed method using benchmarks and a real-world problem.

1 Introduction

Problems of simultaneously optimizing multiple conflicting objective functions are called multi-objective optimization. In recent years, multi-objective evolutionary algorithms (MOEAs) have attracted attention as effective multi-objective optimizers that are able to obtain various trade-off solutions in a single run [1]. The problems in which the real value is used for decision variables are called multi-objective function optimization (MOFO). MOFO is important as it frequently appears in real-world problems. In MOFO, it is worth considering the landscape of not only the objective space but also the variable space as several researchers have pointed out [2,3].

In MOFO problems, multi-modal objective functions produce multiple *local Pareto optima* [4] in the variable space in many cases. When multiple local Pareto optima exist, it becomes harder to find the Pareto optimal set. The problems that have multiple local Pareto optima are called *multi-modal problems*. To find the global Pareto optimal set on a multi-modal problem, MOEAs are required to concentrate the population on promising areas where the Pareto optimal set exists by overcoming local Pareto optima in the early stage of search. We call, in this paper, this part of search process *global search*.

In the last stage of search, after overcoming local Pareto optima, MOEAs are required to uniformly distribute the population over the whole Pareto front in the objective space. We call this search process *local search*. If the population has converged on partial regions of the Pareto front as a result of global search, local search should spread the population along the Pareto front. Here, if the Pareto optimal set gets curved in the variable space, it becomes more difficult for MOEAs to efficiently spread the population along the Pareto front. The Pareto optimal set is curved by several reasons such as epistasis among parameters.

As described above, MOEAs are required to perform conflicting behaviors, converging and spreading the population, depending on the phase of search process. NSGA-II [5] and SPEA2 [6] are recent MOEAs that reportedly obtain relatively good results in various test problems [5,6] and are most widely used today. They are designed to converge the population with *ranking* and spread the population with *sharing* in the survival selection. However, it has been reported that they fail to find Pareto optimal solutions on some multi-modal problems [5,6]. It is also observed that their search efficiency deteriorates on the curved Pareto optimal sets when they are used with crossover operators such as UNDX-m [7], SPX [8] and PCX [9], which show good performance in function optimization.

In this paper, we propose a new MOEA named the Functional-Specialization Multi-Objective real-coded Genetic Algorithm (FS-MOGA), which adaptively switches two search strategies specialized for global and local search. In section two, we first present the formulation of MOFO problems and point out the problems of NSGA-II and SPEA2. In section three, we propose FS-MOGA to overcome these problems. In section four, the performance of FS-MOGA is examined through experiments with benchmark problems. In section five, FS-MOGA and conventional methods are applied to a real-world problem to demonstrate the usefulness of FS-MOGA. Section six states the conclusions.

2 Multi-Objective Function Optimization and Problems of Conventional Approaches

2.1 Multi-Objective Function Optimization

Let the dimensions of the real-valued variable space and the objective space be N and M, respectively. Denote a solution by $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)^T \in \mathbb{R}^N$, the vector of objective functions by $\boldsymbol{f} = (f_1, f_2, \ldots, f_M)^T$, the feasible region by $S \subset \mathbb{R}^N$, and the image of $\boldsymbol{x}$ in the objective space by $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M$. *Multi-objective function optimization problems* can be formulated as:

$$\text{Minimize } f_i(\boldsymbol{x}) \ (i = 1, 2, \ldots, M), \text{ subject to } \boldsymbol{x} \in S.$$

If the following holds for some solutions $\boldsymbol{x_1}, \boldsymbol{x_2} \in S$, $\boldsymbol{x_1}$ is said to be *superior* to $\boldsymbol{x_2}$, which is denoted by $\boldsymbol{x_1} \succ \boldsymbol{x_2}$:

$$\forall i \in \{1, \ldots, M\},\ f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2) \wedge \exists i \in \{1, \ldots, M\},\ f_i(\boldsymbol{x}_1) < f_i(\boldsymbol{x}_2).$$

If there is no feasible solution $\boldsymbol{x}'$ such that $\boldsymbol{x}' \succ \boldsymbol{x}$, the solution $\boldsymbol{x}$ is called a *Pareto optimal solution*. There are often multiple Pareto optimal solutions. The set of all the Pareto optimal solutions is called *Pareto optimal set*. If there is no solution $\boldsymbol{x}'$ such that $\boldsymbol{x}' \succ \boldsymbol{x}$ in the feasible ε-vicinity of a solution $\boldsymbol{x}$, $\boldsymbol{x}$ is called a *local Pareto optimal solution*. Local Pareto optimal solutions henceforth denote those that are *not* Pareto optimal. The image of the Pareto optimal set on the objective space is called a *Pareto front*.

2.2 Problems of Conventional Approaches

We believe that NSGA-II has the following three problems from the viewpoints of global and local search.

High Selection Pressure in Global Search. The aim of global search is to find the areas where the Pareto optimal set exists in the early stage of search. For the global search, it is important to keep the diversity of the population to prevent from trapping in local Pareto optima. NSGA-II chooses parents for crossover with the tournament selection, where the superiority between two individuals is determined by the crowded-comparison operator (CCO) [5]. CCO first compares the ranks of two individuals, and if they are the same, next it compares their crowding distances. Because there are individuals with various ranks in the population in the early stage of search, the superiority between almost all the individual pairs would depend on their ranks. Consequently, individuals on local Pareto optimal regions, which would be highly ranked in the early stage of search, are likely chosen as parents and offspring are sampled from the local Pareto optimal regions intensively. Furthermore, in the survival selection, NSGA-II selects the best μ individuals (μ: population size) from the combined set of the current population and offspring to create a population for the next generation. We believe that this selection would remove individuals from promising areas where individuals have not been sampled sufficiently yet and, as the result, the areas where the Pareto optimal set exists might be missed.

Limited Extrapolative Sampling Ability in Local Search. The aim of the local search is to distribute the population over the whole Pareto front uniformly in the last stage of search. The population often converges on partial regions of the Pareto front as a consequence of global search. In this case, local search is required to spread the population along the Pareto front in the objective space. For this purpose, parents for crossover should be chosen from the areas of the variable space corresponding to those of the objective space where the density of the population distribution is low as shown in Fig. 1. As described above, NSGA-II chooses parents with the tournament selection based on CCO. Because the ranks of almost all the individuals in the population are one in the last stage of search, the superiority between two individuals in the tournament selection primarily depends on their crowding distances. However, since the tournament selection chooses two candidate individuals for a parent from the population uniform-randomly at first, it frequently chooses parents from the dense areas as shown in Fig. 1. If a crossover operator such as UNDX-m, SPX and PCX, which

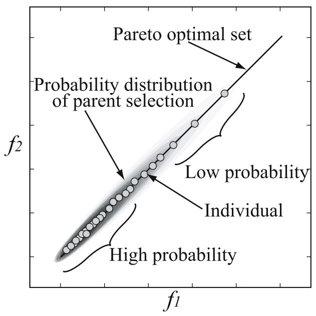


Fig. 1. The probability distribution of parents chosen by the tournament selection in the objective space in mating selection of NSGA-II

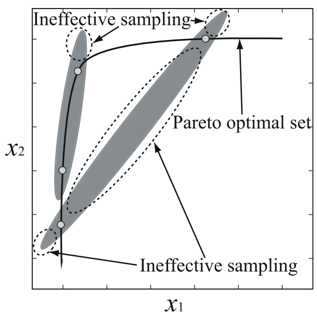


Fig. 2. Offspring distributions generated by parents chosen by the tournament selection in the variable space in NSGA-II

shows good performance in function optimization, is applied to parents in the dense areas, it generates offspring in the same dense areas again at high rates, which means that the search efficiency is lessen.

Low Accuracy of Sampling along the Curved Pareto Optimal Set in Local Search. If the geometrical shape of the Pareto optimal set in the variable space is curved, it is difficult for NSGA-II with crossover operators such as UNDX-m, SPX and PCX that have good search abilities on function optimization to distribute the population along the curved Pareto optimal set. As shown in Fig. 2, the tournament selection used in NSGA-II often mates distant parents on the Pareto optimal set. Unfortunately, the above crossovers with such distant parents generate offspring off the curved Pareto optimal set as shown in Fig. 2.

The basic framework of SPEA2 is the same as that of NSGA-II. In SPEA2, parents are selected with the tournament selection, and the next-generation population is constructed from a combined set of the current population and offspring. SPEA2 determines the superiority between two individuals depending on their strength and distance from the k-th nearest neighbor. We believe that SPEA2 has almost the same problems as NSGA-II. In fact, the observed performances are almost the same [6].

3 Functional-Specialization Multi-Objective Real-Coded Genetic Algorithm: FS-MOGA

In this section, we propose the Functional-Specialization Multi-Objective real-coded Genetic Algorithm (FS-MOGA) to remedy the three problems of the conventional methods described in the previous section. FS-MOGA aims to search efficiently by adaptively switching two strategies, each of which is specialized for global and local search, respectively.

3.1 Global Search Strategy

In the mating selection of the global search, parents for crossover are chosen randomly from the population. This selection resolves the problem of NSGA-II in which highly-ranked individuals on local Pareto optimal regions are frequently chosen as parents. Besides, in the survival selection of the global search, parents in the population are replaced with only their direct descendant. In order to sample new solutions as many as possible, the parents which participate in crossover are always replaced with their offspring like (μ, λ)-ES except that the parents are non-dominated individuals in the population. This alternation resolves the problem of NSGA-II in which individuals on promising areas that have not been searched sufficiently are removed. We believe that these two devices can keep enough diversity of the population and converge the population on the promising areas including the Pareto optimal set better than NSGA-II. In this paper, we employ SPX [8] as a crossover operator of global search. The superiority between two individuals is determined by CCO, like NSGA-II.

3.2 Local Search Strategy

In the mating selection, first, one parent is chosen among the non-dominated individuals in the population according to the probability proportional to the crowding distance [1]. We believe that this selection method enhances the extrapolative sampling ability along the Pareto front in the objective space because it frequently chooses parents on the sparse areas as shown in Fig. 3. Next, the rest of the parents for crossover are chosen randomly from k-nearest neighbors (k-NN) of the first parent in the variable space. This enables to generate offspring properly along the curved Pareto optimal set as shown in Fig. 4. We employ UNDX-m [7] that can efficiently search along the m-dimensional manifold when the population distributes on the m-dimensional manifold. This is because the Pareto optimal set of MOFO with N variables and M objectives is known to form locally $m = \min\{N, M-1\}$ dimensional manifold in the variable space [10]. The survival selection chooses the best μ individuals from the combined set of the current population and the offspring set to create the population of the next generation as NSGA-II does, where μ is the population size. The superiority between two individuals is determined by CCO. This intends to eliminate individuals on dense areas to make the population distribution in the objective space uniformly along the Pareto front.

3.3 Switching Strategies

FS-MOGA chooses a strategy for the current generation according to the ratio of the non-dominated solutions in the population. The probability that an individual randomly chosen from the population is a non-dominated one is equal to the ratio of the non-nominated solutions in the population. Hence, if the rank of a

[1] In this paper, the selection probability of individuals with the crowding distance of ∞ is set to that of the individual with the second largest crowding distance.

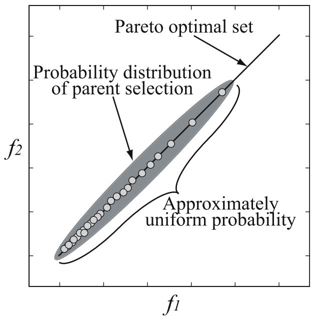

Pareto optimal set
Probability distribution of parent selection
Approximately uniform probability
f2
f1

Fig. 3. The probability distribution of the first parent chosen by the roulette selection in the objective space in mating selection of FS-MOGA *local search*

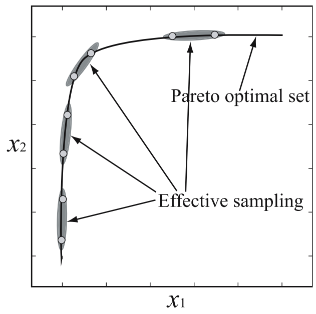


Fig. 4. Offspring distributions generated by parents chosen based on k-NN in the variable space in FS-MOGA *local search*

randomly chosen individual from the population is one, FS-MOGA performs local search. Otherwise, it does global search. Generally, we cannot know whether the region where the population converges is truly Pareto optimal or not. However, we believe that it is desirable to change the strategy from global search to local one because the increase of non-dominated individuals in the population means that the potential to overcome local Pareto optima has lost.

3.4 Algorithm

The algorithm of FS-MOGA consists of three parts; the main loop, the global search procedure and the local search procedure. The global and local search procedures are called from the main loop.

Main Loop

1. *Initialization:* Generate an initial population $P_0 = \{p_1, \ldots, p_\mu\}$, where μ is the population size. Set the generation number $t = 0$.
2. *Evaluation:* Evaluate all the individuals in P_t.
3. *Fitness assignment:* Calculate ranks and crowding distances of all the individuals in P_t.
4. *Strategy selection:* Choose an individual p_{sel} from P_t at random. If p_{sel} is a non-dominated solution ($rank(p_{\mathrm{sel}}) = 1$), then execute **the local search procedure**. Otherwise, perform **the global search procedure**.
5. *Termination check:* Stop if stopping criterion is satisfied. Otherwise, set $t = t + 1$ and go to 3.

Global Search Procedure

1. *Mating selection:* Choose $N + 1$ parents $Q_t = \{q_1, \ldots, q_{N+1}\}$ from P_t randomly without replacement, where N is the number of decision variables.
2. *Crossover:* Apply SPX to Q_t and generate offspring $O_t = \{o_1, \ldots, o_\Lambda\}$, where Λ is the number of offspring generated in global search.

Table 1. Benchmark problems

Problem	N	Objective functions	Bounds	Optimal solutions
RS	40	$f_1(\boldsymbol{x}) = \sum_{i=1}^{N-1}\left[(1-x_i)^2 + 100(x_{i+1}-x_i^2)^2\right]$	$[-5.12, 5.12]^N$	$x_2, \ldots, x_N = \sqrt{x_1}$,
		$f_2(\boldsymbol{x}) = \sum_{i=1}^{N} x_i^2$		$0 \leq x_1, \ldots, x_N \leq 1$
$\mathrm{RR}^{-2.56}$	40	$f_1(\boldsymbol{x}) = \sum_{i=1}^{N-1}\left[(1-x_i)^2 + 100(x_{i+1}-x_i^2)^2\right]$	$[-2.56, 7.68]^N$	$x_2, \ldots, x_N = \sqrt{x_1},\ f_2(\boldsymbol{x}) \leq N$,
		$f_2(\boldsymbol{x}) = 10N + \sum_{i=1}^{N}\left[x_i^2 - 10\cos(2\pi x_i)\right]$		$0 \leq x_1, \ldots, x_N \leq 1$

3. *Evaluation and fitness assignment:* Evaluate all the offspring in O_t. Then, calculate ranks and crowding distances of all the individuals in $P_t \cup O_t$.
4. *Survival selection:* Select the best $N+1$ individuals as survivors $S_t = \{s_1, \ldots, s_{N+1}\}$ from $F_t = Q_t^* \cup O_t$, where F_t has been ordered as follows;
   1) Non-dominated offspring $O_t^* = \{o \in O_t \mid rank(o) = 1\}$,
   2) Non-dominated parents $Q_t^* = \{q \in Q_t \mid rank(q) = 1\}$,
   3) The rest of offspring $\bar{O_t^*} = O_t \setminus O_t^*$.

   where each of O_t^*, Q_t^* and $\bar{O_t^*}$ has been sorted by CCO.
5. *Alternation:* Let $P_{t+1} = (P_t \setminus Q_t) \cup S_t$.

Local Search Procedure

1. *Mating selection:* Choose a parent q_1 from the non-dominated solutions of the population $P_t^* = \{p \in P_t \mid rank(p) = 1\}$ by the roulette-wheel selection based on crowding distances [2]. Then, choose parents $q_2, \ldots, q_{M+1}$ from k-nearest neighbors of q_1 randomly without replacement, where M is the number of objectives and k is a user-defined parameter to determine the range of the neighborhood. Let $Q_t = \{q_1, \ldots, q_{M+1}\}$.
2. *Crossover:* Apply UNDX-m ($m = M$) to Q_t and generate offspring $O_t = \{o_1, \ldots, o_\lambda\}$, where λ is the number of offspring generated in local search.
3. *Evaluation and fitness assignment:* Evaluate all the offspring in O_t. Let $U_t = P_t \cup O_t$. Then, calculate ranks and crowding distances of all the individuals in U_t.
4. *Survival selection:* Sort U_t by CCO. Select the best μ individuals as survivors $S_t = \{s_1, \ldots, s_\mu\}$ from U_t.
5. *Alternation:* Let $P_{t+1} = S_t$.

4 Experiment with Benchmark Problems

To show the effectiveness of FS-MOGA, we compared the performance of FS-MOGA with that of conventional methods on two benchmark problems, RS and $\mathrm{RR}^{-2.56}$, shown in Table 1. RS is a single-modal problem with a curved Pareto optimal set. We use this problem to examine the performance of local search.

[2] The selection probability of individuals assigned crowding distance of ∞ is equal to the one which is the second largest crowding distance.

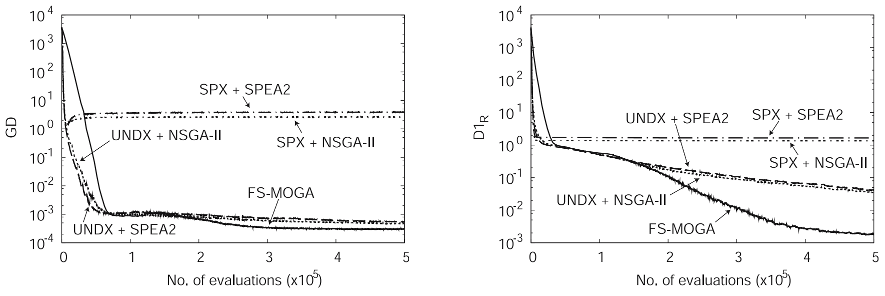


Fig. 5. GD (left) and $D1_R$ (right) on RS averaged over ten trials

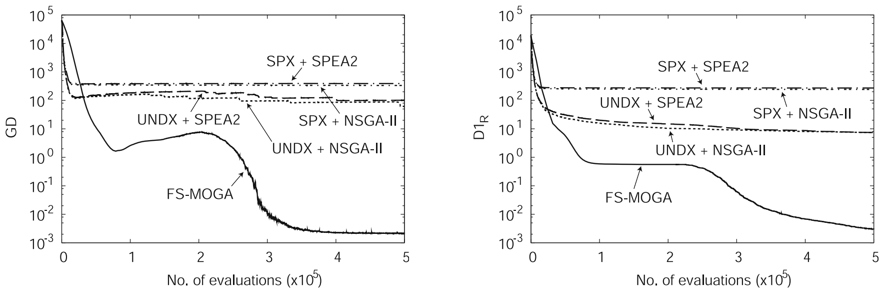


Fig. 6. GD (left) and $D1_R$ (right) on $RR^{-2.56}$ averaged over ten trials

$RR^{-2.56}$ is a multi-modal problem with a curved Pareto optimal set. This problem is used to test the performance of global search and the reasonability of the timing of switching strategies. Because the number of objectives is two in the both problems, their Pareto optimal sets form one-dimensional manifolds in the variable space as described in section 3.2. We therefore employed UNDX (UNDX-1) as a crossover operator in local search of FS-MOGA. We compared the performance of FS-MOGA with that of four conventional methods; NSGA-II with SPX (SPX+NSGA-II), NSGA-II with UNDX (UNDX+NSGA-II), SPEA2 with SPX (SPX+SPEA2) and SPEA2 with UNDX (UNDX+SPEA2). The parameter ϵ of SPX was set to $\sqrt{N+2}$ as its proposers recommended [8]. The population size in each method was set to 400. The number of offspring generated by crossover in global search and local search was 400 and 8, respectively in FS-MOGA, and 8 in the conventional MOEAs. The k-value of k-NN was set to 80, which is 20% of the population size. We have empirically confirmed that the setting of k-value is not sensitive; it would almost always work well with 20% of the population size. The above settings were determined based on the preliminary experiments. We performed ten trials for each method. Each algorithm was stopped when the number of evaluations reached 5×10^5. The performance was measured by two commonly used metrics; GD [1] and $D1_R$ [11]. The smaller the GD is, the nearer the solutions are to the Pareto front on average. The smaller the $D1_R$ is, the wider Pareto front is covered.

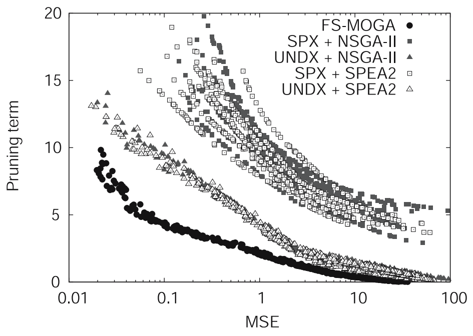


Fig. 7. Result on inference of gene networks. The smaller the MSE and the pruning term are, the better they are.

The results on RS are shown in Fig. 5. As seen from this figure, FS-MOGA outperforms all the conventional methods with respect to GD and $D1_R$. In the $D1_R$ plot in Fig. 5, the improvement rate of $D1_R$ of FS-MOGA is much larger than those of the conventional MOEAs after about 1.2×10^5 evaluations. We confirmed that, after 1.2×10^5 evaluations, all the individuals in the population become non-dominated in FS-MOGA search, which means that FS-MOGA always chooses local search as a search strategy. It suggests that the local search ability of FS-MOGA is better than that of conventional methods.

The results on $RR^{-2.56}$ are shown in Fig. 6. The GD plot in Fig. 6 indicates that FS-MOGA succeeded in finding the Pareto optimal set while the conventional MOEAs failed to overcome local Pareto optima. It suggests that the global search ability of FS-MOGA outperforms that of the conventional methods. On the other hand, the $D1_R$ plot in Fig. 6 shows that FS-MOGA succeeded in spreading the population along the Pareto front widely and uniformly. Therefore, we can conclude that the timing of switching strategies is appropriate.

5 Experiment with a Real-World Problem

To demonstrate the effectiveness of FS-MOGA in a real-world problem, we applied FS-MOGA and the conventional methods, SPX+NSGA-II, UNDX+NSGA-II, SPX+SPEA2 and UNDX+SPEA2, to *inference of gene networks* which is known as a difficult real-world problem. This problem can be formulated as a bi-objective function optimization problem. The first objective is to minimize the mean squared error (MSE) between time course data observed in experiments and those obtained by solving the S-system model [12] which is simultaneous differential equations [12]. The second objective is to minimize the pruning term [13] which represents the complexity of the network. Its decision variables are the system parameters of S-system. We inferred the network consisting of five genes, in which the number of decision variables is 40.

The population size of each method was set to 1,600. The k-value of k-NN used in FS-MOGA was set to 320, which is 20% of the population size. Ten trials were carried out for each method. Each algorithm was stopped when the number of evaluations reached 2×10^6. The other settings were the same as section 4.

The results are shown in Fig. 7. In this figure, the horizontal and vertical axes are MSE and pruning term, respectively. As shown in Fig. 7, FS-MOGA succeeded in finding many better solutions than those found by the conventional methods. We believe that this result attributes the outstanding local and global search abilities of FS-MOGA.

6 Conclusions

In this paper, we proposed a new MOEA for multi-objective function optimization named FS-MOGA, which performs global and local search adaptively. We also showed that FS-MOGA outperformed the conventional methods, SPX+NSGA-II, UNDX+NSGA-II, SPX+SPEA2 and UNDX+SPEA2, on benchmark functions and inference of gene networks.

We think that, in many-objective problems, the strategies switching method proposed in section 3.3 does not work well because most individuals in the population become non-dominated quickly. As future work, an extention of FS-MOGA with ϵ-dominance [14] can be considered for many-objective problems. We also have a plan to compare FS-MOGA with more existing methods and analyze its behavior in detail using problems having three or above objectives and various metrics such as binary quality indicators [15].

References

1. Deb, K., Kalyanmoy, D.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc., New York (2001)
2. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. Evol. Comput. 10(5), 477–506 (2006)
3. Iorio, A.W., Li, X.: Rotationally invariant crossover operators in evolutionary multi-objective optimization. In: Wang, T.-D., Li, X.-D., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 310–317. Springer, Heidelberg (2006)
4. Harada, K., Ikeda, K., Kobayashi, S.: Hybridization of genetic algorithm and local search in multiobjective function optimization: Recommendation of GA then LS. In: GECCO 2006, pp. 667–674. ACM, New York (2006)
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
6. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), pp. 95–100 (2002)

7. Kita, H., Ono, I., Kobayashi, S.: Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. In: CEC 1999, pp. 1581–1587 (1999)
8. Tsutsui, S., Yamamura, M., Higuchi, T.: Multi-parent recombination with simplex crossover in real coded genetic algorithms. In: GECCO 1999, pp. 657–664 (1999)
9. Ballester, P.J., Carter, J.N.: An effective real-parameter genetic algorithm with parent centric normal crossover for multimodal optimisation. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 901–913. Springer, Heidelberg (2004)
10. Hillermeier, C.: Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach. International Series of Numerical Mathematics, vol. 25. Birkhäuser Verlag, Basel (2001)
11. Knowles, J.D., Corne, D.W.: On metrics for comparing non-dominated sets. In: CEC 2002, pp. 711–716 (2002)
12. Tominaga, D., Koga, N., Okamoto, M.: Efficient numerical optimization algorithm based on genetic algorithm for inverse problem. In: GECCO 2000, pp. 252–258 (2000)
13. Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., Tomita, M.: Dynamic modeling of genetic networks using genetic algorithm and S-system. Bioinformatics 19(5), 643–650 (2003)
14. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evol. Comput. 10(3), 263–282 (2002)
15. Zitzler, E., Thiele, L., Laumanns, M., Foneseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. Evol. Comput. 7(2), 117–132 (2003)

Investigations into the Effect of Multiobjectivization in Protein Structure Prediction

Julia Handl, Simon C. Lovell, and Joshua Knowles

The University of Manchester, UK
{j.handl,simon.lovell,j.knowles}@manchester.ac.uk

Abstract. Physics-based potential energy functions used in protein structure prediction are composed of several energy terms combined in a weighted sum. 'Multiobjectivization' — splitting up the energy function into its components and optimizing the components as a vector using multiobjective methods — may have beneficial effects for tackling these difficult problems. In this paper we investigate the hypotheses that multiobjectivization can (i) reduce the number of local optima in the landscapes, as seen by hillclimbers, and (ii) equalize the influence of different energy components that range over vastly different energy scales and hence usually swamp each other's search gradients. The investigations use models of two real molecules, the alanine dipeptide and Metenkephalin under the Amber99 energy function, and consider hillclimbers with a range of mutation step sizes. Our findings support the hypotheses and also indicate that multiobjectivization is competitive with alternative methods of escaping local optima.

1 Introduction

The accurate prediction of protein structure from sequence remains one of the biggest challenges in computational biology [1,10,19]. Recent work has suggested tackling the problem by decomposing the traditional physics-based energy function into two or more energy components, and optimizing the resulting multiobjective function using multiobjective EAs [3,4,18]. The principal argument offered for the attraction of this multiobjective approach is the observation of conflicts between some of the energy components in physics-based energy functions and the fact that an ensemble of candidate solutions rather than a single structure may be obtained [3]. In other words, these papers argue that the set of Pareto optimal solutions, taken as an ensemble, is likely to provide a better answer to the problem of protein structure prediction than would the single-objective optimum, usually a single structure.

In this paper, we are interested in a different aspect of multiobjective optimization, namely the way a decomposition of the energy function impacts on the difficulty of the fitness landscape 'seen' by an optimization method. This is closely related to previous work on 'multiobjectivization' [2,9,12], which argues that the introduction of additional objectives, or the decomposition of an objective into several, may influence the difficulty of a problem, making it easier [2,9,12,14,17] or harder [2]. The approach taken in this paper is an empirical one in which single- and multiobjective hillclimbers present themselves as useful tools to investigate changes in the difficulty of a fitness landscape caused by a decomposition of the energy function. Such an empirical analysis is useful, as general results about the changes in the fitness landscape only directly apply to multiobjective algorithms without archives [7], whose use is rarely practicable in real problems.[1] Also, a straightforward visualization of the multiobjective landscape is not possible even for a two-dimensional problem, as the Pareto dominance relation provides us with a partial ranking of solutions only.

The remainder of the paper is structured as follows. Section 2 discusses the properties of physics-based potential energy functions and the motivations behind their decomposition, in terms of facilitating search. Section 3 discusses the main methods used in this paper, including the two molecular structures considered and the hillclimbers used to explore the resulting fitness landscapes. Experimental results are presented and discussed in Sections 4, 5 and 6. Section 7 considers the wider implications of these results and concludes.

2 Decomposition of Physics-Based Potential Energy Functions

A prototypical physics-based potential energy function (here, Amber99 [5]) can be written as a linear combination of six terms:

$$E_s = E_{bs} + E_{ab} + E_{it} + E_{ta} + E_{vdw} + E_{cc},$$

where E_{bs}, E_{ab}, E_{it} and E_{ta} are the bonded terms constraining bond lengths, bond angles, improper torsion angles and torsion angles respectively. E_{vwd} and E_{cc} are the non-bonded forces, which arise from van der Waals attractive and repulsive forces and electrostatic interactions respectively. E_s is to be minimized. A decomposition into non-bonded and bonded components then considers a two-dimensional vector

$$E_v = (E_{vdw} + E_{cc}, E_{bs} + E_{ab} + E_{it} + E_{ta})^T,$$

rather than a single energy value. The set of solutions that are optimal with respect to E_s form a subset of those that are Pareto optimal with respect to E_v, so minimization of E_v as a Pareto multiobjective optimization problem ([6], page 24) is a valid means of finding a solution to E_s.

The fitness landscapes described by physics-based potential energy functions are highly rugged (multi-modal), which makes them very challenging to optimize. In addition, the scale of the variation within the different energy components differs strongly in these functions: the variation in the non-bonded energies (especially the van der Waals term) is several orders of magnitude larger than that of the bonded terms. Evidently, a large variation in a given term implies the existence of large local gradients in the same terms, which are bound to dominate the overall energy gradients in many areas of the search space. A distinct effect of a decomposition of the function into bonded and non-bonded components is, therefore, an increase in the influence of the bonded objective in those areas of the search space, as the differences in the scales are annihilated and

[1] In particular, [7] shows that multiobjectivization by decomposition causes the introduction of plateaus of incomparable solutions, which can only result in the removal but not in the introduction of local optima in the search space.

the influence of bonded and non-bonded terms is effectively equalized. Importantly, the same effect cannot easily be obtained through a scaling of the individual energy components, as this would not guarantee to preserve the actual energy minimum. The bonded term is smoother than the non-bonded term (as well as having a smaller energy range), so amplifying its influence may help the search process.

The above observation raises the question of whether an increased influence of the bonded components is something that is actually desirable during protein structure prediction. This question can partly be answered through consideration of relevant work in protein structure prediction. Several state-of-the-art prediction methods use mechanisms to suppress the dominating influence of non-bonded energies during the early stages of the search. These measures range from the reformulation or capping of van der Waals forces [19] to a division of forces into short- and long-range components, where long-range components are only periodically updated [8]. The very existence of such techniques suggests that increased guidance by means of bonded terms is seen as favorable at least by some authors.

3 Methods

The Alanine Dipeptide. The alanine dipeptide is a well-known model system in the protein structure prediction literature [15], with only two degrees of freedom. Despite the simplicity of the peptide, its energy landscape already exhibits some fundamental features of the energy landscape of proteins, such as their multimodality and the dominant influence of non-bonded energies. Its small dimensionality, allows for extensive experimental testing and enabled us to visualize directly the (single-objective) energy landscape, algorithm trajectories and the location of local optima during the interpretation of experiments. Due to space limitations these visualizations are not included in the paper.

To create a model of the peptide suitable for optimization, the molecular modeling software TINKER [16] was used to enumerate all possible integer values (from -179 to 180) for the two dihedral angles, and to determine the potential energy of the resulting conformation using the Amber99 force field.

Metenkephalin. The molecule Metenkephalin was used as an example of a more complex molecular structure. This protein consists of five amino-acids and has seventeen flexible dihedral angles, which correspond to the degrees of freedom or decision variables in our problem. A complete enumeration of the search space, as done for the alanine peptide, is no longer possible for this size of problem. Each evaluation therefore requires an explicit call to the TINKER molecular modeling software, making these experiments much more expensive, computationally. As a result, the global optimum for this molecule under the Amber99 energy function was not explicitly identified.

Algorithms. Three different hillclimbers were used to explore the fitness landscapes under integer coding using a standard Gaussian creep mutation operator[2]. The single-objective hillclimber (SHC) always accepts the mutant solution if its objective is equal

[2] We take the floor of the value to make it an integer.

to or better than that of the parent solution. The multiobjective hillclimber (MHC) uses the basic mechanisms described in [13]. It maintains an archive of non-dominated solutions to avoid degradation of solutions (see [13]) and always accepts the mutant solution if it is indifferent or incomparable to the current solution and if it is not dominated by a solution in the archive. The third algorithm, a hybrid hillclimber (HHC), uses single-objective optimization but maintains an archive (of non-dominated solutions under the biobjective formulation) and switches to multiobjective optimization whenever it has failed to find a valid move for 20 consecutive iterations. It switches back to single-objective optimization as soon as an improvement upon the minimum energy value so far has been found.

Experimental Details. In all our experiments (see Sections 4, 5, and 6) all three algorithms were run from identical starting positions with a standard mutation rate of $\frac{1}{n}$, where n is the number of decision variables, and for 10000 iterations. The multiobjective and hybrid hillclimbers used a large archive size of 1000 in order to simulate an unbounded archive and remove any influence of the archive's performance on the search. All experiments were repeated from different starting positions 100 times for the alanine peptide and (due to the much larger computational costs) 15 times for the Metenkephalin molecule. Means of the minimum energy value found per run are reported, and standard errors and p-values (obtained using the Wilcoxon paired rank sum test) are included, where appropriate.

4 Comparative Performance of the Three Hillclimbers

Figure 1 shows the performance of the three hillclimbers for the alanine dipeptide as a function of the standard deviation σ of the Gaussian mutation operator. For $\sigma \geq 35$, all three methods show reliable convergence to the global optimum of -16.98 indicating that escape from all local optima is possible using this size of mutation operator. The results also suggest that very large mutation sizes do not hinder convergence for any of the algorithms, but this is likely to be an artifact resulting from the small size of the search space for this particular problem.

Distinct differences between the algorithms can be observed in the regime for $\sigma <$ 35. For this range of mutation sizes, the single-objective hillclimber converges to local optima with the highest frequency. There are two possible explanations for this result:

1. The hillclimbers utilizing multiobjective optimization can escape local optima more readily at a smaller mutation step size through the exploitation of plateaus of incomparable solutions. Analysis of the trajectories of the single-objective and hybrid hillclimbers provides some evidence for the validity of this latter explanation: the set of solutions accessed by the single-objective hillclimber is usually a subset of those accessed by the hybrid hillclimber. In other words, the trajectories of the two usually agree until a local optimum is met, where the single-objective hillclimber remains while the hybrid hillclimber switches to Pareto optimization and escapes (results not shown).
2. The multiobjective formulation (in particular the larger emphasis on the bonded objective) provides better guidance in the search space and steers the multiobjective algorithm towards the global optimum and away from the local optima. The

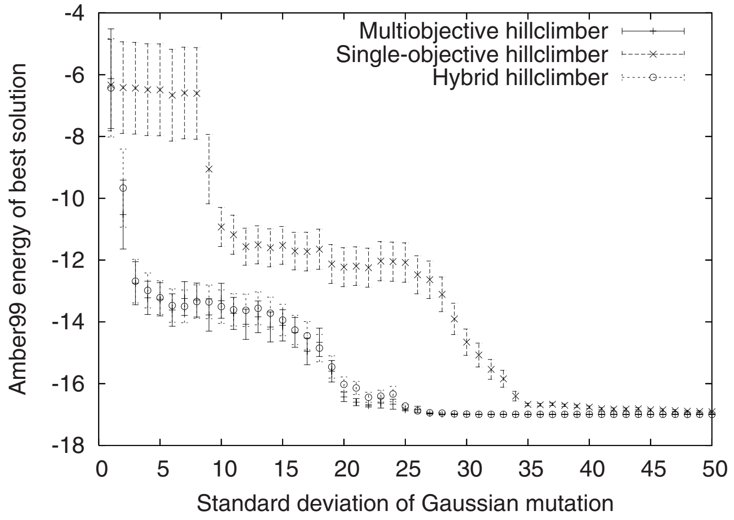


Fig. 1. Performance of the three hillclimbers as a function of the standard deviation σ of the Gaussian mutation operator on the alanine dipeptide. Shown are the averages and the standard error over 100 runs.

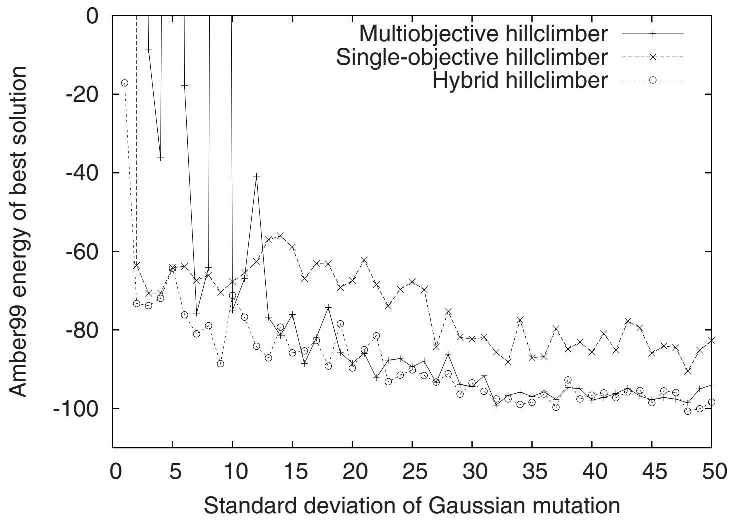


Fig. 2. Performance of the three hillclimbers as a function of the standard deviation σ of the Gaussian mutation operator on Metenkephalin. Shown are the averages over 15 runs. For $\sigma < 15$, SHC and HHC outperform MHC with p-values of 0.0001370 and 3.073e-08, respectively. SHC and HHC are not significantly different at the 0.01 level. For $\sigma \geq 15$, MHC and HHC outperform SHC with a p-value $< 2.2e - 16$. MHC and HHC are not significantly different at the 0.01 level.

good performance of the hybrid hillclimber (which does not utilize multiobjective optimization during the early stages of the search) gives some indication that this is not happening. Nevertheless the validity of this explanation will be investigated further in the next subsections.

In cases where several hillclimbers find the same local or global optimum, the single-objective and hybrid hillclimber are usually more efficient at converging towards the optimal solution than their multiobjective counterpart (results not shown). This is a further side-effect arising from the introduction of plateaus of non-dominated solutions, which cause the multiobjective hillclimber to spend time exploring regions away from the global optimum. For Metenkephalin, this slower convergence actually results in a performance advantage of the single-objective and hybrid hillclimber for small mutation sizes, as shown in Figure 2. Overall, however, a performance advantage of the hillclimbers using multiobjective optimization also remains for this more complex molecule and can now mainly be observed for larger mutation step sizes.

5 Random Decompositions

The above experiments indicate a performance advantage of hillclimbers utilizing multiobjective optimization. As mentioned above, one may speculate that, in addition to the presence of plateaus facilitating the escape from local optima, these methods may benefit from the stronger emphasis on the bonded objective, which may help to direct the search towards the global optimum (and away from local optima).

In order to test this hypothesis further, a set of control experiments were conducted on the alanine dipeptide that compared the performance of a number of alternative decompositions of the overall energy function. In particular, we considered biobjective formulations of the form:

$$F = (E_{bs} + E_{ab} + E_{it} + E_{ta} + E_{vdw} + E_{cc} - r, r)^T .$$

Evidently, E_v is a special case of such a decomposition, where $r = E_{bs} + E_{ab} + E_{it} + E_{ta}$. The alternative definitions of r considered were:

1. r has the same properties as the bonded objective, but is uninformative. This effect was obtained by choosing r to correspond to the bonded energy for a conformation with interchanged phi and psi angles.
2. r is extremely rugged. This effect was obtained by choosing r to correspond to the bonded energy with the phi and psi angles randomly permuted (a random mapping).
3. r presents a smooth gradient (in arbitrary direction). This effect was obtained by choosing r as the sum of all decision variables.

Figure 3 compares the performance of the multiobjective hillclimber on the alanine dipeptide using these alternative decompositions to that of the original multiobjective and single-objective hillclimber. The results confirm that, on the alanine dipeptide, the second objective acts as an escape mechanism: the degree of information provided by r has very little impact and the performance of the hillclimbers is primarily influenced by the ruggedness of r, as a smooth gradient along r provides the facility to ‘drift’

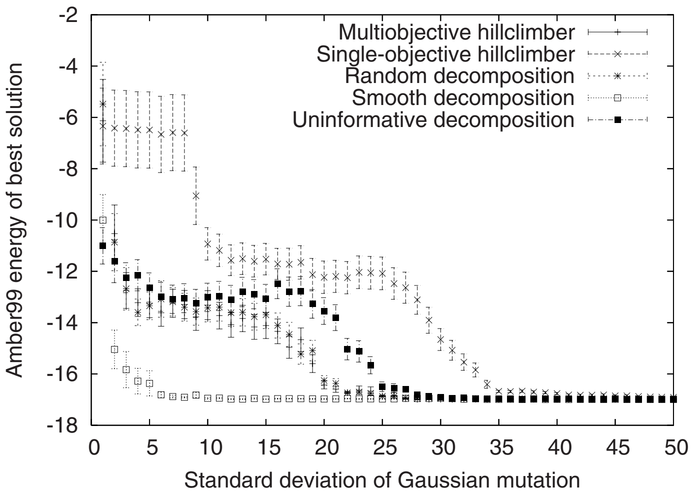


Fig. 3. Performance of SHC and MHC, as well as MHC using three alternative decompositions, as a function of the standard deviation σ of the Gaussian mutation operator on the alanine dipeptide. Shown are the averages and the standard error over 100 runs.

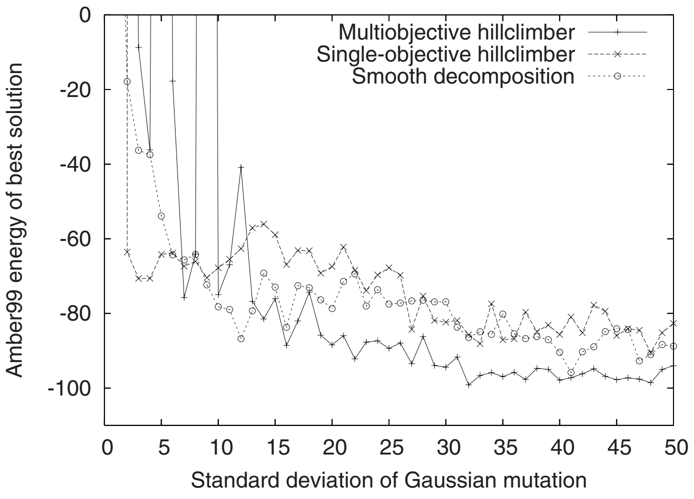


Fig. 4. Performance of SHC and MHC, as well as MHC using the smooth decomposition, as a function of the standard deviation σ of the Gaussian mutation operator on the Metenkephalin molecule. Shown are the averages over 15 runs. For $\sigma < 15$, SHC outperforms MHC and the smooth decomposition with p-values of 0.0001370 and 1.234e-05, respectively. MHC and the smooth decomposition are not significantly different from each other at the 0.01 level. For $\sigma \geq 15$, MHC outperforms SHC and the smooth decomposition with a p-value $< 2.2e - 16$. HHC and the smooth decomposition are not significantly different at the 0.01 level.

out of a local optimum in the first objective. Consequently, the decomposition based on a smooth r turns out as the strongest performer in this comparison and was further evaluated for the Metenkephalin molecule (see Figure 4). For small mutation step sizes ($\sigma \leq 15$), the experimental results on Metenkephalin appear to confirm those obtained for the alanine dipeptide and the decomposition based on the smooth r performs somewhat more robustly than the original decomposition (result not statistically significant). However, for $\sigma \geq 15$ the original decomposition shows a significant performance advantage over the smooth decomposition, indicating that the method may, after all, benefit from the additional guidance provided by the bonded objective. Together with the good performance of the hybrid hillclimber, this result may indicate that increased emphasis on the bonded objectives mainly matters during the escape from local optima.

6 Alternative Escape Mechanisms

Some of the success of the multiobjective and hybrid hillclimber can be attributed to the introduction of plateaus that facilitate the escape from local optima. On the downside, the presence of these plateaus slows down convergence speed, which is at the root of the superior performance of the single-objective and hybrid hillclimbers on Metenkephalin for small mutation step sizes. There is thus a trade-off to be met regarding the introduction of plateaus, and it is likely that more effective escape mechanisms than multiobjectivization exist.

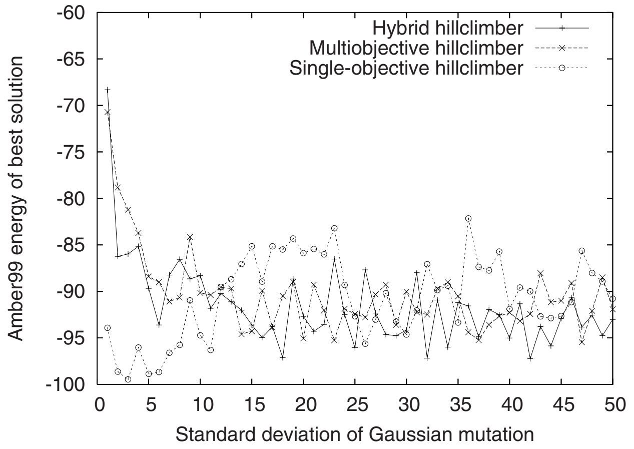


Fig. 5. Performance of the three hillclimbers with macromutation as a function of the standard deviation σ of the Gaussian mutation operator on the Metenkephalin molecule. Shown are the averages and the standard error over 15 runs. For $\sigma < 15$, SHC outperforms MHC and HHC with p-values of 4.534e-14 and 7.731e-11, respectively. MHC and HHC are not significantly different from each other at the 0.01 level. For $\sigma \geq 15$, MHC, SHC and HHC are not significantly different at the 0.01 level.

In a final experiment, we consider the relative performance of the three hillclimbers if they are furnished with additional escape mechanisms. In particular, the mechanism chosen here is the macromutation operator proposed in [11], which simulates uniform crossover between the current and a random solution. The offspring that inherits more than fifty per cent of its genes from the current solution is taken as the mutant solution. This macromutation is applied with a probability of 0.7 in every iteration.

Figure 5 shows the results obtained for Metenkephalin. The results obtained show a significant increase in the performance of the single-objective hillclimber for small mutation step sizes. In contrast, the performance of the multiobjective and hybrid hillclimbers suffers from the introduction of the macromutation, (which may appear surprising given their increased performance for large mutation step sizes — see Figure 2). This is probably because, for the multiobjective hillclimbers, the macromutation causes them to spend too much time in plateaus (of non-dominated solutions) in the search space, which affects the degree of convergence that can be achieved by them.

When comparing the results obtained using the best parameter settings for each of the three types of hillclimbers (best overall averages are obtained by the single-objective hillclimber with macromutation for $\sigma = 4$, the hybrid hillclimber without macromutation with $\sigma = 48$ and the multiobjective hillclimber without macromutation with $\sigma = 32$), no statistically significant difference can be observed.

7 Conclusion

This study has explored the impact of multiobjectivization on the potential energy functions used in protein structure prediction. Compared with a simple hillclimber, a multiobjective hillclimber operating on a decomposed two-objective energy function finds lower overall energy solutions for the same number of evaluations - and does so over a range of mutation step sizes. Experiments to investigate this advantage indicate that multiobjectivization achieves a reduction in the number of local optima in the landscape whilst simultaneously maintaining some of the important “guidance” (or gradient) that the landscape possesses.

When comparing multiobjectivization to more advanced search methods, namely the inclusion of a well-respected macromutation operator to facilitate escape from local optima, we find no advantage in terms of the minimal energy achieved at the best mutation step-size settings. However, the multiobjective approach seems slightly more robust to different step-size choices. More importantly, multiobjectivization finds the low energy solutions at the same time as finding many other non-dominated trade-off solutions, and at no extra cost in function evaluations. Whether or not these additional trade-offs are valuable for identifying native structures is not considered here, but is the subject of our future work.

Acknowledgments. JH gratefully acknowledges support by a Special Training Fellowship from the Medical Research Council (MRC), UK. JK is supported by a David Phillips Fellowship from the Biotechnology and Biological Sciences Research Council (BBSRC), UK.

References

1. Bonneau, R., Tsai, J., Ruczinski, I., Chivian, D., Rohl, C., Strauss, C., Baker, D.: Rosetta in CASP4: progress in ab initio protein structure prediction. Proteins (suppl. 5), 119–126 (2001)
2. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do additional objectives make a problem harder? In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 765–772. ACM Press, New York (2007)
3. Cutello, V., Narzisi, G., Nicosia, G.: A multi-objective evolutionary approach to protein structure prediction. J. R. Soc. Interface 3(6), 139–151 (2006)
4. Day, R.O., Zydallis, J.B., Lamont, G.B., Pachter, R.: Solving the protein structure prediction problem through a multiobjective genetic algorithm. Nanotechnology 2, 32–35 (2002)
5. Duan, Y., et al.: A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. Journal of Computational Chemistry 24(16), 1999–2012 (2003)
6. Ehrgott, M.: Multicriteria Optimization. Springer, Berlin (2005)
7. Handl, J., Lovell, S., Knowles, J.: Multiobjectivization by decomposition of scalar cost functions. In: Rudolph, G., et al. (eds.) PPSN X 2008. LNCS, vol. 5199, pp. 31–40. Springer, Berlin (2008)
8. Jacobson, M., Pincus, D., Rapp, C., Day, T., Honig, B., Shaw, D., Friesner, R.: A hierarchical approach to all-atom protein loop prediction. Proteins: Structure, Function, and Bioinformatics 55(2), 351–367 (2004)
9. Jensen, M.: Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation. Journal of Mathematical Modelling and Algorithms 3(4), 323–347 (2004)
10. Jones, D.: Predicting novel protein folds by using FRAGFOLD. Proteins (Suppl. 5), 127–132 (2001)
11. Jones, T.: Crossover, macromutation, and population-based search. In: Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 73–80. Morgan Kaufmann, San Francisco (1995)
12. Knowles, J., Watson, R., Corne, D.: Reducing local optima in single-objective problems by multi-objectivization. In: Proceedings of the Congress on Evolutionary Multiobjective Optimization, pp. 269–283. Springer, Berlin (2001)
13. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)
14. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Natural Computing 5(3), 305–319 (2006)
15. Ramachandran, G.N., Ramakrishnan, C., Sasisekharan, V.: Stereochemistry of polypeptide chain configurations. Journal of Molecular Biology 7, 95–99 (1963)
16. Ren, P., Ponder, J.W.: Polarizable atomic multipole water model for molecular mechanics simulation. Journal of Physical Chemistry B 107, 5933–5947 (2003)
17. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. Journal of Mathematical Modelling and Algorithms 3(4), 346–366 (2004)
18. Schulze-Kremer, S.: Application of evolutionary computation to protein folding with specialized operators. In: Fogel, G.B., Corne, D.W. (eds.) Evolutionary Computation in Bioinformatics, pp. 163–191. Morgan Kaufmann, San Francisco (2003)
19. Simons, K., Kooperberg, C., Huang, E., Baker, D.: Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. Journal of Molecular Biology 268, 209–225 (1997)

On the Use of Projected Gradients for Constrained Multiobjective Optimization Problems

Alfredo G. Hernandez-Diaz[1], Carlos A. Coello Coello[2], Luis V. Santana-Quintero[2], Fatima Perez[3], Julian Molina[3], and Rafael Caballero[3]

[1] Department of Economics, Quantitative Methods and Economic History, Pablo de Olavide University, Seville, Spain
agarher@upo.es

[2] Centro de Investigacion y de Estudios Avanzados, Mexico D.F., Mexico
ccoello@cs.cinvestav.mx, lvspenny@hotmail.com

[3] Department of Applied Economics(Mathematics), University of Malaga, Malaga, Spain
f_perez@uma.es, julian.molina@uma.es, rafael.caballero@uma.es

Abstract. Recent works have shown how hybrid variants of gradient-based methods and evolutionary algorithms perform better than a pure evolutionary method both for single-objective and multiobjective optimization. This same idea has been used with Evolutionary Multiobjective Optimization (**EMO**), obtaining also very promising results. In most cases, gradient information is used as part of the mutation operator (and only for unconstrained MOPs), in order to move every generated point to the exact Pareto front. In our approach, we use the Karush-Kuhn-Tucker optimality condition for constrained optimization problems to combine the information provided by the gradient vector of each objective function and the gradient vectors of constraint functions to obtain a feasible movement direction in those points near the border. In our approach, gradients of the objective functions will be approximated using quadratic regressions, trying to avoid local optima. The proposed algorithm is able to converge on several nonlinear constrained multiobjective optimization problems obtained from a benchmark, consuming few objective function evaluations (between 150 and 1000). Our results indicate that our proposed scheme may produce a significant reduction in the computational cost, while producing results of good quality, when it is incorporated into a hybrid MOEA or when it is used to seed an EMO algorithm.

Keywords: Gradient-based method, constrained optimization, nonlinear multiobjective programming, quadratic approximation.

1 Introduction

MOEAs have been very successful in the solution of a wide variety of problems, mainly during the last few years [2]. However, for certain types of applications, MOEAs result particularly expensive (computationally speaking), since they require a large number of objective function evaluations in order to produce an acceptable approximation of the true Pareto front, specially for constrained problems, where a suitable constraint-handling mechanism is needed.

On the other hand, the classical (exact) methods for (multi-objective) optimization (gradient based methods) consume just a few number of evaluations, but can be trapped in local optima and require a lot of assumptions about the problem: continuity, differentiability, explicit mathematical formulation, etc.

On the other hand, under proper assumptions, Newton's method is quadratically convergent, but its efficiency is reduced by its expensive computational cost, especially, for mid-to-large scale problems. The key point is to evaluate the gradient and the Hessian efficiently, and two different approaches can be found for that sake:

– **Use analytical derivatives.** The first option is manually obtaining analytic derivatives of each objective and constraint functions. But this is only possible if an explicit mathematical formulation is available, and this is the main weakness of this approach as many interesting problems could not be solved: simulation based problems, design problems, etc. On the other hand, it is an error-prone activity, because if the formulation is complicated, obtaining analytical derivatives can be a hard task.
– **Use estimated derivatives.** In this category we can find the Newton-like methods, where derivatives are estimated in some efficient way. These methods do not require explicit formulae of derivatives but, on the other hand, consume some more evaluations in order to compute the estimation.

On the other hand, the use of gradient information for constrained optimization problem has been relatively popular for many years. Techniques such that Barrier Methods and Penalty functions (see [3]), Interior-Point Methods ([7]) or Projected Gradient ([5], [8]) have been successfully used in continuous optimization.

Barrier and penalty methods are designed to face the problem by solving a sequence of specially constructed unconstrained optimization problems. In a penalty method, the feasible region is expanded from the feasible region to all of $\mathbb{R}^n$, but a large cost or "penalty" is added to the objective function for points that lie outside of the original feasible region. In a barrier method, we assume we are given a starting point in the interior of the feasible region, and we impose a very large cost on feasible points that lie ever closer to the boundary of the feasible region, thereby creating a "barrier" to exiting the feasible region. Interior-point methods move through the interior of the feasible region following the gradient vector of the objective function generating approximated solutions that asymptotically converge to the exact solution, while the projected gradient orthogonally projects new generated infeasible solutions over the feasible region.

In this work we propose an easy-to-implement method to iteratively generate nondominated solutions for constrained multiobjective optimization problems. In this method we **use "global" estimated derivatives** for the objective functions (and **analytical derivatives** for constraint functions) but consuming the lowest possible number of evaluations while maintaining a high quality on the results. We propose its use to seed and EMO method instead of using it along the whole process (which would consume too many evaluations).

2 Definitions and Basic Concepts

We assume the following definition of a constrained MOP problem[1]:

$$\text{Minimize } \boldsymbol{f}(\boldsymbol{x}) := (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_s(\boldsymbol{x})) \tag{1}$$

subject to:

$$g_i(\boldsymbol{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{2}$$

$$h_i(\boldsymbol{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ is the vector of decision variables (normally bounded $a_i \leq x_i \leq b_i$), $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., s$ are the objective functions, and $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$, $j = 1, ..., p$ are the continuously differentiable constraint functions of the problem.

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, for $x \in \mathbb{R}^n$, a direction $v \in \mathbb{R}^n$ is a *descent direction* if $\nabla f(x) \cdot v < 0$ (∇f denotes the gradient vector of f).

A generalized gradient method can be summarized in the following equation:

$$x^{k+1} = x^k + \alpha^k v^k$$

where v^k is a descent direction and α^k is the step size. One of the most commonly used choice for the descent direction is the following (*steepest descent direction*):

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

Obviously, one of the main difficulties for constrained problems is the feasibility of x^{k+1}. Specially when the constraints are nonlinear, a balance has to be achieved between satisfying the constraints and reducing the objective function.

Moreover, choosing the optimum step size α^k is desirable, but it may be computationally expensive. Some other approaches, which have good properties (e.g., convergence), are quite efficient. One of the most efficient is the Armijo's rule:

Let $\beta \in (0, 1)$ be a prespecified value, let v be a descent direction and let x be the current point. The condition to accept t (the step size) is:

$$f(x + tv) \leq f(x) + \beta t \nabla f(x) \cdot v$$

where we start with $t = 1$ and while this condition is not satisfied we set $t := t/2$.

The choice of β can be critical because the bigger the value of β, the bigger the steps we can implement at the beginning; but we consume more evaluations if too many reductions of t must be done to achieve the condition. Armijo's rule is mathematically correct and the "t" value always exists. However, this value could

[1] Without loss of generality, we will assume only minimization problems.

be very small, which would be translated into an insignificant progress (this is, in fact, the main disadvantage of Armijo's rule). This problem is more significant for box-constrained problems or, in general, for constrained problems when the current solution is over or close to the boundary between the feasible and infeasible regions, or to the boundary of one of the decision variables, and the descent direction moves it outside of the feasible space. Depending on the violated constraints, we distinguish two cases: (a) if the new solution x^{k+1} violates constraints in (2) or (3) or, (b) if some variables of x^{k+1} are out of its range.

(a) Let denote by $c_1(x), c_2(x), ..., c_q(x)$ the violated constraints by x^{k+1}. The Karush-Kuhn-Tucker optimality condition (for equality constraint problems) states that x^* is a local minimizer if there exist real numbers $\lambda_1, ..., \lambda_q$ (Lagrange multipliers) such that

$$\nabla f(x^*) = \sum_{i=1}^{q} \lambda_i \nabla c_i(x^*).$$

In a regular situation (for example, if $\nabla f(x^*)$ and $\nabla c_i(x^*), i = 1, ..., q$, are independent), the above condition is imposible to achieve. This means there exists a feasible direction v obtained by decomposing $\nabla f(x^*)$ in its projection over the space generated by $\{\nabla c_1(x^*), ..., \nabla c_q(x^*)\}$ and its normal component, v. So, this normal vector v is computed taken into account that $v = \nabla f(x^*) - \sum_{i=1}^{q} \lambda_i \nabla c_i(x^*)$ has to be orthogonal to $\nabla c_i(x^*)$, for all i. So, the coefficient vector $(\lambda_1, ..., \lambda_q)$ may be obtained by solving the system (all gradient vectors are evaluated in x^*)

$$\begin{pmatrix} \nabla c_1 \cdot \nabla c_1 & \nabla c_1 \cdot \nabla c_2 & \cdots & \nabla c_1 \cdot \nabla c_q \\ \nabla c_2 \cdot \nabla c_1 & \nabla c_2 \cdot \nabla c_2 & \cdots & \nabla c_2 \cdot \nabla c_q \\ \vdots & \vdots & \vdots & \vdots \\ \nabla c_q \cdot \nabla c_1 & \nabla c_q \cdot \nabla c_2 & \cdots & \nabla c_q \cdot \nabla c_q \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_q \end{pmatrix} = \begin{pmatrix} \nabla f \cdot \nabla c_1 \\ \nabla f \cdot \nabla c_2 \\ \vdots \\ \nabla f \cdot \nabla c_q \end{pmatrix}.$$

Then, if x^{k+1} violates constraints $c_1, c_2, ..., c_q$ and the current solution x^k is close enough to them (in order to consider these constraints as active constraints), the feasible direction considered is the above normal vector (reducing the step size until x^{k+1} is feasible). The key issue is the following: The closer x^k to these violated constraints, the more precise the feasible direction v but, due to some of these constraints are nonlinear, the closer x^k to the constraints, the smaller the step size to obtain a feasible move. In our experiments, x is considered close, or ε-active, to constraint c_i if $\frac{|c_i(x)|}{\nabla c_i(x) \cdot \nabla c_i(x)} < \varepsilon$. ($\varepsilon = 0.001$ for linear constraints and $\varepsilon = 0.1$ for nonlinear constraints).

(b) In this case, we apply the following rules for each violated variable i:

- If $x_i^{k+1} < a_i$, then $x_i^{k+1} = \frac{a_i + x_i^k}{2}$.
- If $x_i^{k+1} > b_i$, then $x_i^{k+1} = \frac{b_i + x_i^k}{2}$.

This way, the current solution x^k is moved in a intermediate direction between the direction induced by $\nabla f(x^k)$ and its projection over the violated constraints (like in (a)). In an informal sense, this kind of transformation is also an adapted Armijo's rule but considering different step sizes in each coordinate.

3 Gradient Based Method for Multi-Objective Optimization

The goal now is trying to adapt some of the principles of single-objective optimization to obtain a number of efficient points of the MOP problem. The main idea is based on the Fritz-John optimality condition for MOP problems (see for example [4]):

– Given a point $x \in X$, a necessary condition to be Pareto optimal solution is the existence of $\boldsymbol{\lambda} \geq 0$ such that $\sum_{i=1}^{p} \lambda_i \nabla f_i(x) = 0$.

For a bi-objective optimization problem, this condition means that for any Pareto optimal solution, we can find some $\lambda \geq 0$ such that $\nabla f_1(x) = -\lambda \nabla f_2(x)$. This is, for any Pareto optimal point, gradients of both objective functions are parallel but in the opposite direction. It means that if we are placed in the minimum of one of the objectives (for example the minimum of f_1, a Pareto optimal solution) and follow the direction of $\nabla f_2(x)$, we will remain in the Pareto front. This is shown graphically in Figure 1.

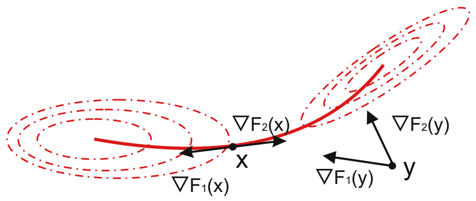


Fig. 1. Pareto front on a bi-objective problem

This idea was used in [10], where they link $s+1$ local searches (more precisely, tabu searches). The first local search starts from an arbitrary point and attempts to find the optimal solution to the problem with the single objective f_1. Let x_1 be the last point visited at the end of this search. Then, a local search is applied again to find the best solution to the problem with the single objective f_2 using x_1 as the initial solution. This process is repeated until all the single-objective problems associated with the s objectives have been solved. At this point, they solve again the problem with the first objective f_1 starting from x_s, to finish a cycle around the efficient set. This phase yields the s efficient points that approximate the best solutions to the single-objective problems that result from ignoring all but one objective function, and additional efficient solutions may be found during this phase because all visited points are checked for inclusion in the approximation of the Pareto front, as probably most of the intermediate points will lie on the Pareto front. This way, they obtain an initial set of efficient points to be used as an initial population for the EMO method developed in [10].

In this work, we are going to use the same idea, link $s+1$ single objective local searches, but using a single-objective gradient based method instead of a tabu search. The next subsection is devoted to show the main features on this gradient-based local search mechanism.

3.1 Single-Objective Gradient Based Method

For our local search engine, we are going to use an steepest descent method, this is, given the current point x^k, the next point will be computed as follows:

$$x^{k+1} = x^k - t \cdot \widetilde{\nabla} f(x^k)$$

where $\widetilde{\nabla} f(x^k)$ is an estimation of $\nabla f(x^k)$ (or its projection/modification seen in the above section), and t will be computed following our adapted Armijo's rule with $\beta = 0.1$ and starting with the value of $t = 1$. The reason to choose a low value for β is the fact that small steps are also interesting for us while we are on the Pareto front, as we are checking every intermediate solution for being included in the final approximation. This is, we are not only interested in the final point of each search, but also in the intermediate points.

To estimate the gradient of a function f, we will use a quadratic approximation over all its domain:

$$f(x) \approx \beta_0 + \sum_{i=1}^{n} \beta_i^1 \cdot x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} \beta_{i,j}^2 \cdot x_i \cdot x_j$$

This means that we are interested in global gradients instead of the local information provided by a precise estimation of the gradients at each solution.

The number of parameters (N) to adjust such an approximation for a function with n variables is: $N = 1 + n + \frac{n(n+1)}{2} = \frac{n^2+3n+2}{2}$. N represents the minimum number of points needed to adjust such an approximation. For a problem with 30 variables, for example, at least 496 will be needed. In order to generate these N points efficiently, we used Latin-Hypercubes [9], which is a method that guarantees a good distribution of the initial population in a multidimensional space, as it is required in order to better fit the function with this quadratic approximation. Once these points are generated and evaluated, we compute the values of each parameter solving the corresponding system of equations using a pseudo-inverse (due to its complexity when N is increased). This system of equations can be formulated using matrices: $X \cdot B = Y$, so $X \cdot B = (X^t X)^{-1} X^t Y$.

Finally, we assumed the following stopping conditions:

1. The step is too small, this is, the estimated gradient or the projected gradient is too small: $t \cdot \|\widetilde{\nabla} f(x_k)\| < 0.01$, or
2. The improvement is too small: $|f(x_{k+1}) - f(x_k)| < 0.001$.

The complete method is summarized in Algorithm 1.

Algorithm 1. Constrained Multi-Objective Gradient Based method: CMGBM

```
1: Generate a set *InitPop* with $N$ initial points using Latin-Hypercubes.
2: Send each point in *InitPop* to the list of nondominated solutions: $PF$.
3: Use the set *InitPop* to adjust a quadratic approximation of each objective function over all its domain.
4: **for** each objective function $f_i$ (repeating the first one) **do**
5:     $x^0$ =*last point visited* or *random solution in* $PF$ *when* $i = 0$
6:     **while** stopping conditions = FALSE or $x^{k+1}$ is infeasible **do**
7:         Obtain $x^{k+1}$ through the gradient-based method using $\widetilde{\nabla} f_i(x^k)$.
8:         If $x^{k+1}$ is infeasible, check $\varepsilon$-active constraints for $x^k$.
9:         If $x^k$ is an interior point (it has no $\varepsilon$-active constraints), reduce the step size. Otherwise, obtain a new $x^{k+1}$ through the projected gradient over the $\varepsilon$-active constraints.
10:        Send $x^{k+1}$ to $PF$.
11:    **end while**
12: **end for**
```

4 Preliminary Results

To test the performance of CMGBM, we solved several constrained optimization problems from the benchmark: Srinivas ([13]), Osyczka and Osyczka2 ([11]), Tanaka ([14]), Binh ([1]) and Jimenez ([6]). All of them are nonlinear bi-objective optimization problems with several nonlinear constraints. Moreover, they all have 2 decision variables, except for Osyczka which has 6 variables. For a quick overview of these test functions, please visit: `http://www.cs.cinvestav.mx/~emoobook/`.

Table 1. *IGD* values for the selected six test problems

Test Function / *IGD*	**Min**	**Mean (SD)**	**Max**	**N. Points**	**N. Eval.**
Srinivas	0.057	0.089 (0.013)	0.107	74.455	406.6
Osyczka	0.171	0.296 (0.136)	0.570	64.000	544.2
Osyczka2	0.102	0.140 (0.018)	0.159	37.546	917.4
Tanaka	0.091	0.489 (0.410)	1.227	17.091	239.8
Binh	0.046	0.072 (0.012)	0.084	224.727	594.1
Jimenez	0.010	0.043 (0.063)	0.216	129.091	262.3

Results obtained by CMGBM are not compare against other algorithms because the main aim is to show the viability of this scheme to obtain nondominated solutions over the true Pareto front. This is why we perform 11 independent runs, measured using the Inverted Generational Distance, *IGD* ([15]). *IGD* measures the euclidean distance from the true Pareto front to the approximated front, previously normalized to allow a fair comparison. So, the closer the *IGD* value to zero, the better the approximation. $IGD = 0$ is obtained only when the approximated front is over the true Pareto front and the extremes have been also achieved.

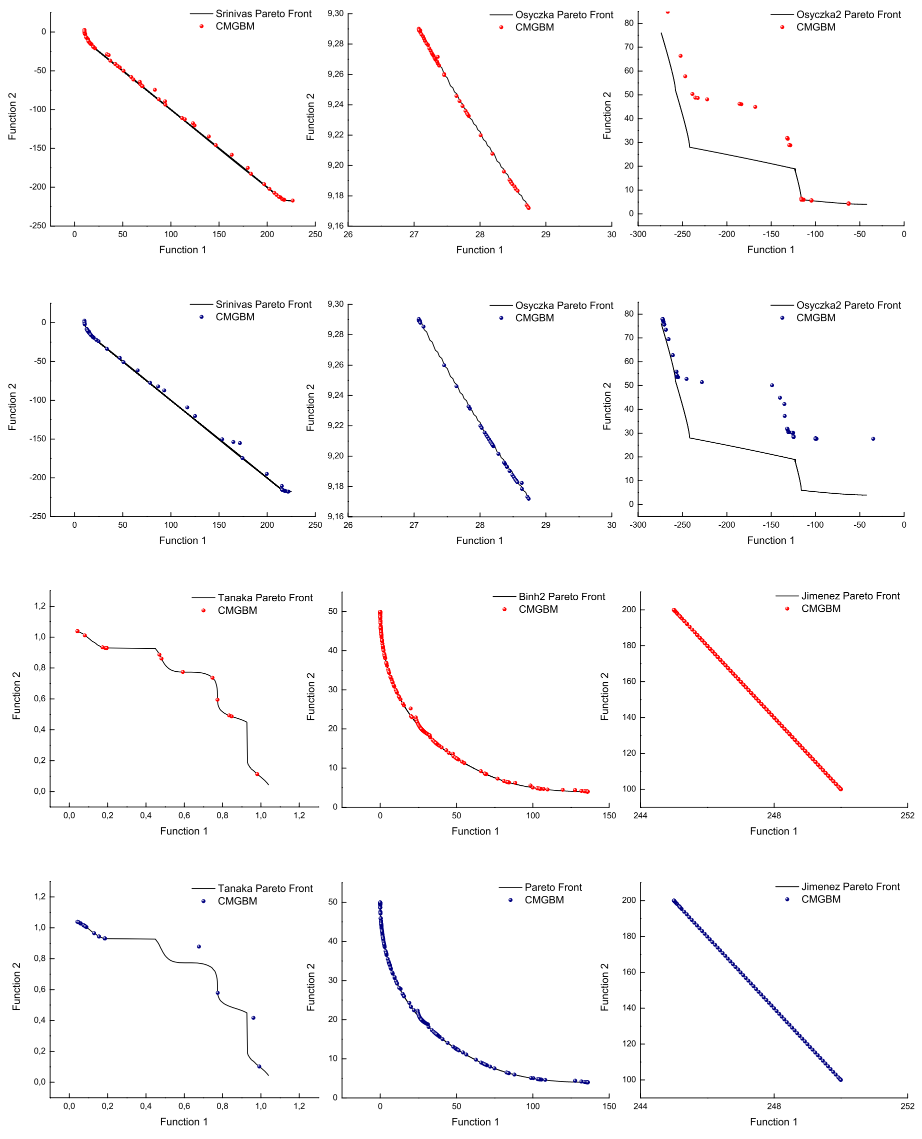


Fig. 2. Best (top) and median (bottom) IGD values obtained by CMGBM for selected problems

It can be observed in Table 1 that CMGBM produced IGD values really close to zero. The first column shows the best IGD values (min), the second column shows the mean IGD values and its corresponding standard deviation. The third column show the worst (max) of the eleven IGD values. Finally, last two columns show the average of the number of nondominated solutions obtained by CMGBM and the mean value of the number of evaluations consumed. For these problems, the CMGBM is able to find a high number of exact efficient points using very few evaluations.

Figure 2 shows the nondominated solutions obtained by CMGBM. These plots correspond to the best run (top) and the run in the median value (bottom) with respect to the *IGD* metric. We can clearly see that in all problems, but Osycka2, CMGBM converged to the true Pareto front after only about 500 fitness function evaluations. And even for the hardest problem (Osyczka2), CMGBM obtained some good solutions after only 1000 evaluations. On the other hand, CMGBM was able to obtain the extreme points (of each objective function) in most cases.

5 Conclusions

We have introduced a Constrained Multi-Objective Gradient-Based Method (CMGBM) in order to generate efficient solutions of nonlinear constrained multi-objective optimization problems with a low number of objective function evaluations. The main contribution is the way in which we use the estimated gradient vector of the objective functions and the gradient vector of the constraints to obtain an improvement direction. Results show the efficiency of this method over several nonlinear constrained MOPs, since CMGBM obtains good approximations of the Pareto front consuming very few objective function evaluations. With this preliminary results we show how the use of gradient information could reduce the computational cost while quality is not decreased. We believe that this gradient information could be very useful to seed EMO algorithms and enhance their convergence. This strategy could be more efficient than using gradients through all the EMO execution because once the EMO method is provided with solutions close (or in) to the Pareto front, the use of gradient information consumes a lot of evaluations while not providing significant improvements.

In the future, besides completing a comprehensive set of experiments, we would like to improve this scheme using also approximated gradient vectors of the constraints, and adapt this mechanism for problems with more than two objective functions.

Acknowledgments

The second author gratefully acknowledges support from CONACyT project 45683-Y. The third author acknowledges support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN. The other authors acknowledge support from the Andalusian Regional Government and the Spanish Ministry of Education and Science.

References

1. Binh, T.T., Korn, U.: MOBES: A multiobjective evolution strategy for constrained optimization problems. In: The Third International Conference on Genetic Algorithms (Mendel 1997), Brno, Czech Republic, pp. 176–182 (1997)
2. Coello Coello, C.A., Lamont, G.B.: Applications of Multi-Objective Evolutionary Algorithms. World Scientific, Singapore (2004)
3. Fiacco, A.V., McCormick, G.P.: Nonlinear Programming. In: Classics Appl. Math., vol. 4, SIAM, Philadelphia (1990); Reprint of the 1968 original
4. Fliege, J., Svaiter, B.: Steepest Descent Methods for Multicriteria Optimization. Mathematical Methods of Operations Research 51(3), 479–494 (2000)
5. Goldstein, A.A.: Convex programming in Hilbert Space. Bulletin of the American Mathematical Society 70, 709–710 (1964)
6. Jiménez, F., Verdegay, J.L., Gómez-Skarmeta, A.F.: Evolutionary Techniques for Constrained Multiobjective Optimization Problems. In: Wu, A.S. (ed.) Proc. of GECCO, Orlando, Florida, pp. 115–116 (1999)
7. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica 4, 373–395 (1984)
8. Levitin, E.S., Polyak, B.T.: Constrained Minimization Problems. USSR Computational Mathematics and Mathematical Physics 6, 1–50 (1966)
9. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2), 239–245 (1979)
10. Molina, J., Laguna, M., Marti, R., Caballero, R.: SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization. INFORMS Journal on Computing 19(1), 91–100 (2007)
11. Osyczka, A., Kundu, S.: A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Structural Optimization 10, 94–99 (1995)
12. Schaffler, S., Schultz, R., Weinzierl, K.: Stochastic method for the solution of unconstrained vector optimization problems. Journal of Opt. Theory and Applications 114(1), 209–222 (2002)
13. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1994)
14. Tanaka, M., Watanabe, H., Furukawa, Y., Tanino, T.: GA-Based Decision Support System for Multicriteria Optimization. In: Proc. of the International Conference on Systems, Man, and Cybernetics, vol. 2, pp. 1556–1561. IEEE, Piscataway (1995)
15. Van Veldhuizen, D.A.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Phd. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)

Diversity Maintenance Mechanism for Multi-Objective Genetic Algorithms Using Clustering and Network Inversion

Tomoyuki Hiroyasu[1], Kenji Kobayashi[2], Masashi Nishioka[2], and Mitsunori Miki[3]

[1] Faculty of Life and Medical Sciences, Doshisha University, 1-3 Tatara Miyakodani Kyotanabe, Kyoto, Japan
[2] Graduate School of Engineering, Doshisha University
[3] Faculty of Science and Engineering, Doshisha University
{tomo@is,kkobayashi@mikilab,mnishioka@mikilab,mmiki@mail}.doshisha.ac.jp

Abstract. One of the major issues in applying multi-objective genetic algorithms to real-world problems is how to reduce the large number of evaluations. The simplest approach is a search with a small population size. However, the diversity of solutions is often lost with such a search. To overcome this difficulty, this paper proposes a diversity maintenance mechanism using clustering and Network Inversion that is capable of preserving diversity by relocating solutions. In addition, the proposed mechanism adopts clustering of training data sets to improve the accuracy of relocation. The results of numerical experiments on test functions and diesel engine emission and fuel economy problems showed that the proposed mechanism provided solutions with high diversity even when the search was performed with a small number of solutions.

1 Introduction

Multi-objective genetic algorithms (MOGAs) are strong optimization methods that can derive a Pareto-optimal set in a single run [1,2]. However, in real-world problems, such as large-scale design problems [3,4], the reduction of evaluation calls becomes an essential issue due to their high computational cost. Two major approaches have been proposed for this issue: the response surface method [5,6,7] and search using small population size [8] (SSP strategy). In this paper, an effective SSP mechanism is discussed. SSP is a simple approach for reducing the calculation cost. However, solution diversity in the objective space tends to be lost. We have proposed a diversity maintenance mechanism using an Artificial Neural Network (ANN) [9] to preserve high diversity, which relocates the converged solutions to solutions with a uniform distribution. In this relocation, it is necessary to perform inverse analysis that estimates the design values from the fixed objective values, because the relocation must be conducted in the objective space. In our previous paper, inverse function by ANN was adopted, but it showed poor performance in high-dimensional or multi-modal problems. In this paper, new diversity maintenance mechanism based on Network Inversion [10,11] and clustering of ANN training data is proposed to solve these difficulties.

2 Mode of Inverse Analysis and Necessity of Clustering

2.1 Mode of Inverse Analysis in the Proposed Mechanism

In the proposed mechanism, Network Inversion (NI) [10,11] is applied for inverse analysis, which is a technique using an approximation function with the same structure as the objective function. This mode is advantageous with regard to high-dimensional problems. In the conventional mode based on inverse function, it is difficult to relocate solutions appropriately, because many outputs should be estimated from few inputs in high dimensions. On the other hand, NI preserves the input/output relationship of the objective function, and is effective in high dimensions. Most studies concerned with ANN focus only on an approximation of the objective function and local search [12,13]. On the other hand, ANN was used in this study to relocate the solutions and restore the solution diversity.

2.2 Necessity of Clustering

The filtration of training data set to improve the approximation accuracy of NI is also discussed. To create a good approximation function with few data, the following two conditions should be satisfied:
(i). Training data set exists in a narrow area.
(ii). Approximation function is a monotone in approximation area.

A case where a function pole exists in a training area is illustrated in Fig.1 to discuss the importance of these conditions.

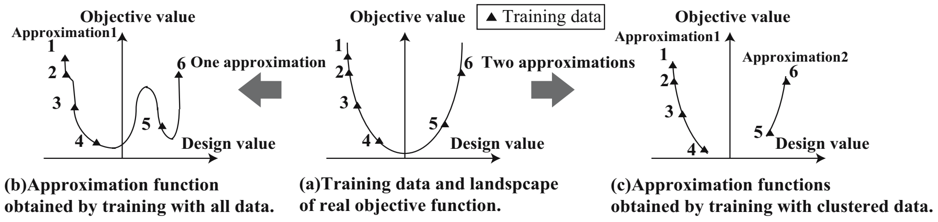

Objective value
Approximation1
1
2
3
4
5
6
One approximation
Design value
(b)Approximation function obtained by training with all data.
Objective value
▲ Training data
Two approximations
(a)Training data and landspcape of real objective function.
Approximation2
(c)Approximation functions obtained by training with clustered data.

Fig. 1. Case where a function pole exists in a training area and its handling

In Fig.1, Fig.1(c) which satisfies both (i) and (ii) by dividing the training data set into two groups, is a good method of training when a function pole exists in the training area. Whether there is a function pole in the training area can be judged according to the neighbor relationship of the data. To check this, it is necessary to make two sorting lists of all data, which are sorted by design values and objective values in ascending order. When two adjacent data points of an arbitrary data set are the same in both lists, the training data set is defined as having the same neighbor relationship. For example, when all training data in

Fig.1(a) are sorted by design and objective values (Fig.1(b)), each list becomes (1,2,3,4,5,6) and (4,5,3,2,6,1). In this case, the data set has a different neighbor relationship. On the other hand, in Fig.1(c), training data are divided into two groups: (1,2,3,4,) and (5,6). If the same sorting procedure is performed on both groups, the neighbor relationship becomes the same. In this paper, we propose a clustering method that can divide training data sets into groups with the same neighbor relationship.

3 Diversity Maintenance Mechanism by Clustering and Network Inversion

The proposed mechanism is composed of MOGA search, clustering, training ANNs, and relocation, and the latter 3 processes are used to restore diversity. The concept of the proposed mechanism is illustrated in Fig.2.

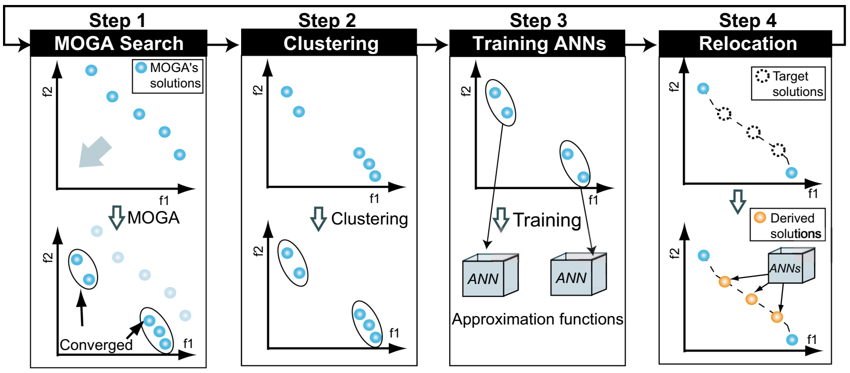


Fig. 2. Concept of proposed mechanism

The first application of the restoration process is when all archive solutions become the non-dominated solutions (NDS), and is applied uniformly in the remaining search. The algorithm is described below:

Step 1: MOGA search. MOGA search is performed until application condition of the diversity restoration is met. After application of the diversity restoration, MOGA search is conducted for specified number of generations.

Step 2: Clustering. Clustering is applied to the NDS obtained by MOGA.

Step 3: Training ANNs. ANNs are trained based on a set of clustered solutions obtained in Step 2.

Step 4-1: Linear interpolation. A linear line passing through a set of n NDS is obtained by interpolation.

Step 4-2: Locating target solutions (TS). All solutions are removed except those at the edges, and $n-2$ TS are located uniformly on the interpolated line.

Step 4-3: Inverse analysis. Inverse analysis is performed with NI, and design values corresponding to each objective value of TS are derived.

Step 4-4: Relocation. Obtained design values are evaluated using the real objective function. Then, archive and solutions obtained by NI are combined, and the archive update mechanism of MOGA is executed. Return to Step 1 if the terminal condition is not satisfied.

3.1 Clustering (Step 2)

A clustering algorithm is proposed to obtain a set of solutions with the same neighbor relationship. It judges the neighbor relationship by calculating the Euclidean distance between one solution and the others. The solution used to calculate the distance is defined as the base solution, and the solutions with minimum or maximum value in any objective function are defined as edge solutions. Furthermore, this operation is applied only to n NDS of archive, because it showed superior results in the preliminary experiment. The clustering procedure is described below, and the clustering of the solutions in Fig.3 is shown in Fig.4

Step 2-1: ID is assigned to k NDS in ascending order regarding f1 $(i = 0)$.

Step 2-2: A base solution is fixed $(i = 0 :$ solution with ID=1, $i = 1 :$ ID=$k)$.

Step 2-3: Euclidean distances between the base solution and the others are calculated in the design space, and a sorted list of IDs is made by the distance in ascending order.

Step 2-4: Consecutive solutions from the head of the sorted list with its IDs in ascending $(i = 0)$ or descending $(i = 1)$ order are selected as setA_i

Step 2-5: If $i = 0$, return to Step 2-2 and update the base solution $(i = 1)$.

Step 2-6: When setA_0=setA_1, a cluster is created from the solutions of set A_0, and the process is terminated. When either set becomes the subset of the other, the solutions of the subset are adopted as a cluster, and the process is terminated. Otherwise, only sets with the edge solution included are selected as the new NDS set, and the clustering process is repeated to it.

This algorithm does not require the number of clusters to be assigned beforehand, because only clusters that include the edge solution are obtained.

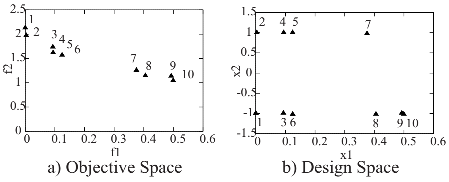


Fig. 3. Distribution of a data set

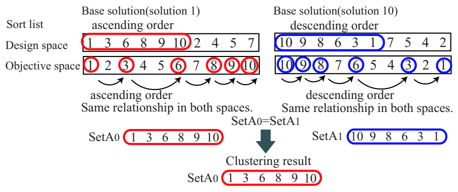


Fig. 4. Concept of clustering

3.2 Training ANNs (Step 3)

In Step 3, training ANNs is performed regarding design values as input and objective values as output by Backpropagation (BP). Solutions from clustering in Step 2 are adopted as a training data set. With BP, output error is minimized based on the gradient method by considering the weights of the network as the source of error and adjusting it.

3.3 Linear Interpolation and Location of Target Solutions (Steps 4-1 and 4-2)

In this section, we describe how the objective values of TS are determined. First, a linear interpolation line is obtained (Step 4-1). For this interpolation, a linear interpolation method is adopted, because it showed more positive results in preliminary experiments than two-dimensional interpolation. Next, TS are located on the interpolated line such that the distances on the interpolated line between the neighboring solutions become the same. As it is difficult to set TS in many objectives (more than three), this paper focuses on two-objective problems. The scheme of locating TS is shown in Fig.5.

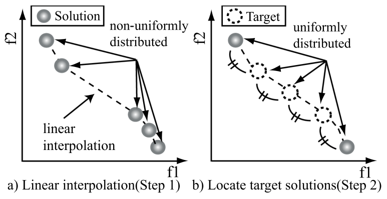

Solution
non-uniformly distributed
f2
linear interpolation
f1
a) Linear interpolation(Step 1)
Target
uniformly distributed
f2
f1
b) Locate target solutions(Step 2)

Fig. 5. Concept of locating TS

3.4 Inverse Analysis (Step 4-3)

In this step, input values are estimated from fixed output values using ANNs trained in Step 3. The principle of inverse estimation by NI is the same as that of BP. However, the source of error is considered to be the input values, and they are adjusted instead of the weights of the network. From the process described above, design values x of TS are estimated. In addition, training cost of ANNs and calculation cost of NI are relatively small in comparison with the evaluation cost of GA, and these costs are not discussed in this paper.

3.5 Relocation (Step 4-4)

Estimated design values from Step 4-3 are evaluated by the objective function, and real-objective values are obtained. Next, the archive update mechanism is applied to a set of solutions composed of solutions of inverse analysis and MOGAs. With this, a superior set of solutions with high accuracy and diversity can be selected. In addition, even if TS cannot be obtained properly, the search performance will not be degraded because the updating mechanism eliminates them and the solutions before relocation are adopted.

4 Numerical Experiments through Test Functions

The effectiveness of the proposed mechanism was verified through numerical experiments. Search performance was compared by Angular Cover Rate (ACR) for diversity and GD for accuracy [14]. In ACR, domains in which solutions exist are divided uniformly into the number of search population size by the angle, and it counts how many domains are covered by derived solutions. On the other hand, GD measures the accuracy by calculating the distance between derived solutions and Pareto-optimal solutions. In all experiments discussed in this section, NSGA-II is employed for MOGAs of the proposed mechanism, and population size is set to 10 [8], because the search with a small number of solutions is assumed. In addition, crossover rate is 1.0, and mutation rate is 1.0/gene length. The concept of ACR and GD is illustrated in Fig.6 and Fig.7.

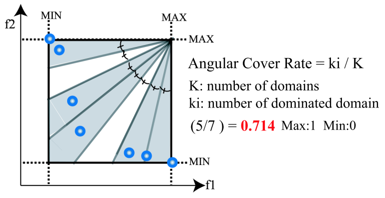

f2
MIN
MAX
MAX
MIN
f1
Angular Cover Rate = ki / K
K: number of domains
ki: number of dominated domain
(5/7) = 0.714 Max:1 Min:0

Fig. 6. Concept of ACR

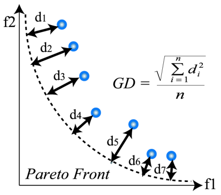


Fig. 7. Concept of GD

In this section, the following three experiments are discussed.

1. Effectiveness of clustering.
2. Improvement of diversity by the proposed mechanism.
3. Diversity maintenance and accuracy by iterating the proposed mechanism.

4.1 Accuracy of an Approximation by Clustering and NI

The differences in performance between NI and NI with clustering were verified using multi-modal function ZDT4 [15] with 2-dimensions. The training data set and the results are shown in Fig.8 and Fig.9, respectively.

The results shown in Fig.9 indicate that when the neighbor relationships in both spaces are different, clustering may provide more accurate approximation.

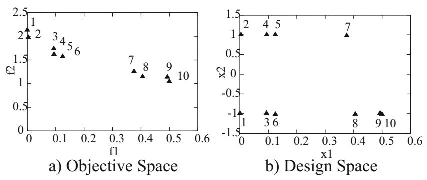


Fig. 8. Test data set

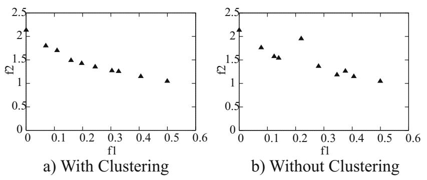


Fig. 9. Effectiveness of NI with clustering

4.2 Effect of the Proposed Mechanism for Diversity

In this experiment, MOGA search and the proposed mechanism were compared to examine the effects of the proposed mechanism on diversity. Test functions were ZDT1, ZDT2, and ZDT4 with three patterns of dimensions (2,5,10), and the number of generations for GA was 100. The results are shown in Fig.10.

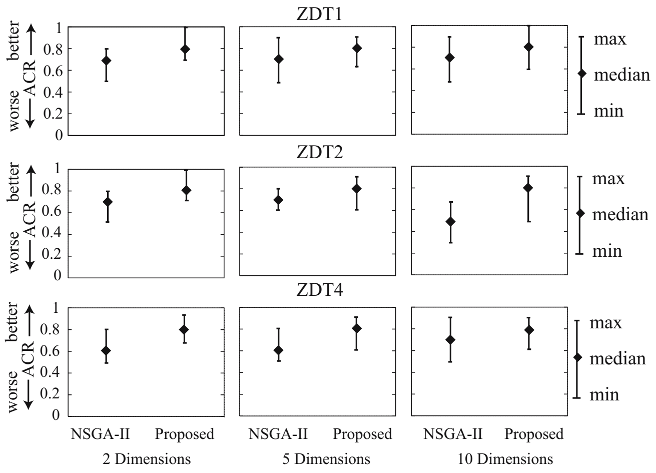


Fig. 10. ACR (Max, Median, Min) in ZDT1, ZDT2, ZDT4

Fig.10 shows the minimum, median, and maximum of ACR at 30 runs. From these results, we can infer that the proposed mechanism provides solutions with high diversity even in high dimensions.

4.3 Effect of the Iteration of the Proposed Mechanism

The effects of diversity maintenance by the proposed mechanism were verified. In ZDT1 and ZDT2, generation was set to 100, and the number of applications of the proposed mechanism was set to 5. In ZDT4, each parameter was set to 300 and 15, respectively. In addition, the number of dimensions was 10. In this experiment, the effects of diversity maintenance by the proposed mechanism were validated by the transition of ACR. The results are shown in Fig.11.

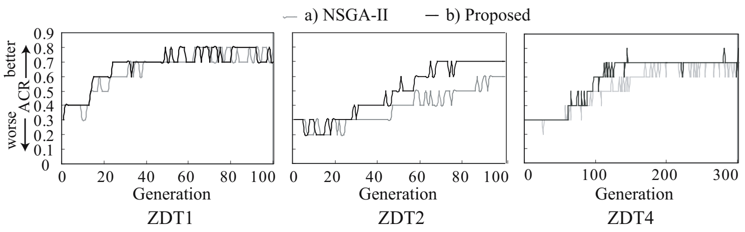


Fig. 11. Transition of ACR in ZDT1,ZDT2,ZDT4

Fig.11 indicates the median of ACR measured by the generation. These results show that the search with the proposed mechanism can maintain high diversity compared to only using MOGAs. Next, the influence regarding the accuracy by the proposed mechanism was verified by GD. The results on median and standard deviation are shown in Table.1. The number of evaluations was the same in both methods.

Table 1. Comparison of GD

		ZDT1	ZDT2	ZDT4
NSGA-II	Median	0.209	0.348	3.53
	Standard deviation	0.106	0.128	1.66
Proposed	Median	0.238	0.362	3.06
	Standard deviation	0.130	0.214	1.31

As shown in Table.1, both methods showed comparable accuracy. From these results, we concluded that the proposed mechanism can maintain solution diversity without deterioration of solution accuracy even in high dimensions.

5 Numerical Experiments through Real-World Problem

The effectiveness of the proposed mechanism was verified by application to the diesel engine emission scheduling problem.

5.1 Diesel Engine Fuel Emission Scheduling Problem

Diesel engines have considerable advantages with regard to durability, fuel economy, and reduced CO_2 emission. However, in recent years, emission regulations for automobile engines have become stricter because of the adverse influence on the environment. Therefore, there is a great deal of research interest in reducing emissions to meet the regulations [16].

A diesel engine works by compressing air in a cylinder and injecting a liquid fuel. However, it also emits large amounts of both nitric oxide (NO_x) and particulate matter (PM), including soot. The amounts of these emissions are determined by the scheduling of fuel emission. Therefore, the emission characteristics were optimized from the perspective of engine combustion to achieve low emission. This optimization problem is called the diesel engine emission scheduling problem, and minimizes the following objectives:

– The amount of NO_x
– The amount of soot
– The amount of specific fuel consumption (SFC)

This problem should be optimized simultaneously due to the trade-off relationships between NO_x and soot or SFC. These objective values are calculated by simulation with the phenomenological model HIDECS [17,18]. Design variables are Start Angle, Exhaust Gas Recirculation Rate (EGR Rate), Swirl Ratio, Boost Pressure, and the shape of injection [19].

In this section, it is examined whether the proposed mechanism can improve the diversity, and also if the diversity can be preserved by iterating them.

5.2 Effects of Relocation in the Diesel Problem

The effects on diversity were verified when the proposed mechanism was applied once. In all of the following problems, only SFC and NOx were focused upon because the mechanism can only be utilized with two-objective problems. Population size was set to 10 and the number of generations was 50. The other GA parameters were the same as in section 4. The results are shown in Fig.12.

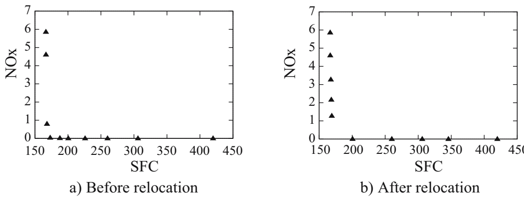


Fig. 12. Effect of the relocation of the proposed mechanism

As shown in Fig.12, the diversity can be improved by the proposed mechanism in the diesel engine problem.

5.3 Effects of Iteration in the Diesel Problem

The effects on diversity maintenance of the proposed mechanism were verified. In this experiment, the transition of ACR was examined during a search. The plot graphs of obtained NDS at 30 runs were compared to measure the accuracy, because Pareto-optimal solutions in this problem were not known and GD could not be employed as an indicator. The number of generations was set to 50 and the number of applications of the proposed mechanism was set to 10. The number of evaluations was the same in both methods. The first experimental results of the transition of ACR are illustrated in Fig.13. Fig.13 shows that the proposed mechanism can provide solutions with higher diversity during a search than the conventional search.

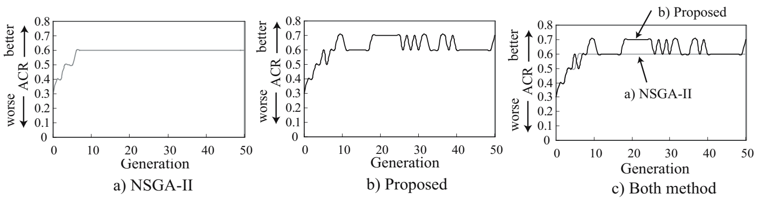


Fig. 13. Transition of ACR in diesel problem

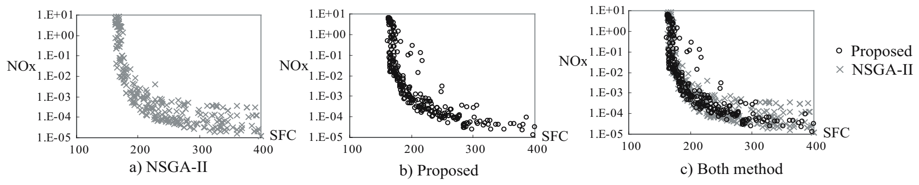


Fig. 14. All NDS obtained by both methods in diesel problem

In the second experiment, the influence of iteration on the search performance was examined by plotting all obtained NDS at 30 runs in Fig.14.

Fig.14 indicates that the search performance of the proposed mechanism is comparable to the conventional method with regard to accuracy. From these results, we concluded that the proposed mechanism can provide solutions with high diversity without deterioration of the search performance in the diesel engine emission scheduling problem.

6 Conclusions and Future Work

In this paper, a new diversity maintenance mechanism was proposed that hybridizes MOGAs with a small population size and the process of restoring diversity by NI. This mechanism involves MOGA search, clustering, training ANNs, and relocation, and diversity can be preserved by iteration.

The effectiveness of the proposed mechanism was examined through application to benchmark problems and the diesel engine emission scheduling problem. The results showed that the proposed mechanism can provide solutions with high diversity even in high dimensions and the search with 10 solutions. In future studies, we will examine the appropriate number of solutions for the search, and how to apply the proposed mechanism to many-objective problems (more than 3).

References

1. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India (2000)
2. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Performance of the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich (2001)
3. Chiba, K., Obayashi, S.: High-Fidelity Multidisciplinary Design Optimization of Aerostructural Wing Shape for Regional Jet. In: 23rd Applied Aerodynamics Conference (2005)
4. Hiroyasu, T., Miki, M., Kamiura, J., Watanabe, S., Hiroyasu, H.: Multi-Objective Optimization of Diesel Engine Emissions and Fuel Economy using Genetic Algorithms and Phenomenological Model. In: SAE, Powertrain and Fluid Systems Conference (2002)

5. Shirai, T., Arakawa, M., Nakayama, H.: Approximate Multi-Objective Optimization Using RBF Network. In: The Computational Mechanics Conference, vol. 18, pp. 759–760 (2005)
6. Peixoto, J.L.: Hierarchical Variable Selection in Polynomial Regression Models. The American Statistician 41(4), 311–313 (1987)
7. Sacks, J., et al.: Design and Analysis of Computer Experiments 4. Statistical Science 4, 409–435 (1989)
8. Obayashi, S., Sasaki, D., Oyama, A.: Finding Tradeoffs by Using Multiobjective Optimization Algorithms. Transactions of JSASS 47(155), 51–58 (2004)
9. Kobayashi, K., Hiroyasu, T., Miki, M.: Mechanism of Multi-Objective Genetic Algorithm for Maintaining the Solution Diversity Using Neural Network. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 216–226. Springer, Heidelberg (2007)
10. Linden, A., Kindermann, J.: Inversion of multilayer nets. In: Proc. Int. Joint conf. on Neural Networks, pp. 425–430 (1989)
11. Ogawa, T., Kaneda, H.: Complex-Valued Network Inversion for Solving Complex-Valued Inverse Problems. Bulletin of science and engineering, Takushoku University 9(4), 83–84 (2006)
12. Cunha, A.G., Vieira, A.: A Hybrid Multi-Objective Evolutionary Algorithm Using an Inverse Neural Network. Hybrid Metaheuristics, 25–30 (2004)
13. Adra, S.F., Hamody, I., Griffin, I., Fleming, P.J.: A Hybrid Multi-Objective Evolutionary Algorithm Using an Inverse Neural Network for Aircraft Control System Design. In: 2005 IEEE Congress on Evolutionary Computation (CEC 2005), vol. 1, pp. 1–8 (2005)
14. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms, pp. 326–327. John Wiley and Sons, Chichester
15. Deb, K., Meyarivan, T.: Constrained Test Problems for Multi-Objective Evolutionary Optimization. KanGAL report 200005, Indian Institute of Technology, Kanpur, India (2000)
16. Aoyagi, Y.: A Survey of Existing Emission Reduction Technology for Gasoline-powered Engine and Future Prospect 55(9), 10–16 (2001)
17. Hiroyasu, H., Kadota, T., Arai, M.: Development and Use of a Spray Combustion Modeling to Predict Diesel Engine Efficiency and Pollutant Emissions (Part 1 Combustion Modeling). Bulletin of the JSME 26(214), 569–575 (1983)
18. Hiroyasu, H., Kadota, T., Arai, M.: Development and Use of a Spray Combustion Modeling to Predict Diesel Engine Efficiency and Pollutant Emissions (Part 2 Computational Procedure and Parametric Study). Bulletin of the JSME 26(214), 576–583 (1983)
19. Itoh, S., Nakamura, K.: Reduction of Diesel Exhaust Gas Emission with Common Rail System 55(9), 46–52 (2001)

Many Objective Optimisation: Direct Objective Boundary Identification

Evan J. Hughes

Department of Informatics and Sensors,
Cranfield University, DCMT, Shrivenham, UK. SN6 8LA
ejhughes@theiet.org

Abstract. This paper describes and demonstrates a new and highly innovative technique that identifies an approximation of the entire bounding surface of the feasible objective region directly, including deep concavities, disconnected regions and the edges of interior holes in the feasible areas. The Pareto front is a subset of the surface of the objective boundary and can be extracted easily. Importantly, if the entire objective boundary is known, breaks and discontinuities in the Pareto front may be identified using automated methods; even with high objective dimensionality. This paper describes a proof-of-principle evolutionary algorithm that implements the new and unique Direct Objective Boundary Identification (DOBI) method.

1 Introduction

The objective boundary is the 'outside hypersurface' of the hypervolume of the feasible objective region in objective space. In its full form, it encompasses both maximisation and minimisation of the objectives as demonstrated in Fig. 1a. The leading edge of the objective boundary that is heading towards a utopia point of interest is the *objective front*. Deep concavities in the objective front may be dominated, but are still viable solutions. The Pareto front is the sub-set of non-dominated solutions, given a criteria of minimisation or maximisation for each objective. If a solution is identified as belonging to the objective front and is a minimal/maximal solution as desired, but is not part of the Pareto front, then the solution must lie in a discontinuity of the Pareto front. Thus regions which are true discontinuities in the Pareto front can be identified, rather than just not knowing if solutions exist, but have not been identified by the Pareto-based optimiser.

Many real engineering problems [1] require 4 or more objectives and optimisation and visualisation of the results becomes difficult. Previous work has shown that identification of spot solutions on the objective front can yield useful information about the structure of the Pareto front [2]. Additionally, any optimisation algorithms that are intended for use in many dimensions must be capable of producing useful front approximations as the problem dimensionality increases [3,4,5].

Regions of the objective space that have no feasible solutions associated are also of interest. An algorithm that is attempting to identify the full objective boundary should be capable of identifying the boundary of disconnected valid objective regions and also identify the boundaries of 'holes' within feasible objective regions.

This paper describes a new and very unique algorithm that is designed to identify the objective boundary directly for many-objective problems. The approximation of the Pareto front can then be extracted from the results and analysed. The prime focus of this research has been to see if a practical algorithm for direct identification of the objective boundary could be demonstrated. The described algorithm can identify the boundaries of convex and concave regions, disjoint regions and also identify the boundaries of large 'holes' within feasible spaces. The algorithm uses a normalisation process to allow the search to be conducted easily in high-dimensional objective spaces and the paper provides a theoretical basis for key elements of algorithm tuning.

2 Objective Boundary Definition

The *objective boundary* is the set of points that form the boundary of the feasible objective region. The *objective boundary set* is the corresponding points in decision space. In objective space, points could be considered as *interior*, where they are surrounded on all sides by other feasible points, or *exterior* which have at least one direction in which they have no immediate neighbouring feasible solutions. Mathematically, the set of *exterior* solutions $\mathcal{E}$ is a subset of the entire feasible objective set $\mathcal{Q}$, $\mathcal{E} \subseteq \mathcal{Q}$, and is defined formally in (1);

$$\mathcal{E} = \{\exists \vec{n} : \vec{\mathcal{E}} + \delta t \vec{n} \nsubseteq \mathcal{Q}\} \tag{1}$$

ie. for each member of the set $\mathcal{E}$ (the objective vector denoted as $\vec{\mathcal{E}}$), there exists at least one direction $\vec{n}$, which when examined over a small distance, δt, there are no solutions in $\mathcal{Q}$ that form part of the feasible objective region. The definition is demonstrated graphically in Fig. 1a where the directions in which no immediate neighbours exist have been indicated.

3 Neighbourhood Assessment

In reality, the objective region is sampled by the optimisation process, rather than being a continuous set around each point of interest. There are many possible approaches to identifying which of the objective-space sample points are surrounded and therefore interior. Common methods for reconstructing convoluted surfaces may be considered, such as Delaunay Triangulation and level sets [6]. Delaunay Triangulation fundamentally is the basis of many of the reconstruction methods and forms a connected net between the observed valid points in the feasible region such that for any simplex in the net (triangle in 2D, tetrahedron in 3D etc.) there are no points within the circumhypersphere of the simplex. The objective surface will form a subset of the faces of the Delaunay triangulation. Unfortunately the computation of the Delaunay triangulation with high numbers of objective dimensions (≥ 10) becomes very time consuming. For computationally and theoretical simplicity, and also to provide a processing time that is linear in the number of objectives, an alternative to hypersphere and Delaunay based methods has been sought.

The concept of the new algorithm described in this paper is to assess the quality of each solution by examining the largest empty *hypercone* (or an approximation to

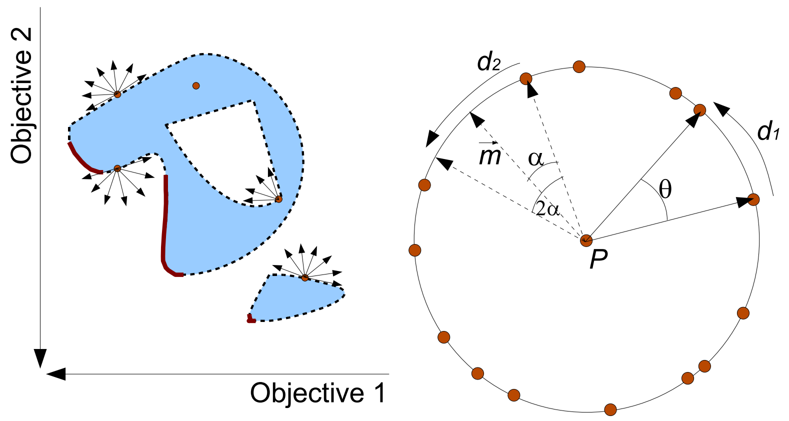


Fig. 1. Left figure shows diagram of feasible objective solutions, showing the boundary that forms the objective front (dashed), and also the Pareto front when the objectives are minimised (Solid). The objective region shown has two disjoint feasible regions, one of which has a 'hole'. The figure shows a selection of feasible solutions and the directions in which there are no close neighbouring solutions. The right figure shows a diagram of ideal conditions where neighbours are distributed uniformly and densely on the surface of a hypersphere. Arc length d_1 and nearest-neighbour angle θ will be distributed exponentially. The angle α is between a test direction, $\vec{m}$, and its nearest neighbour and α will also be distributed exponentially, but with half the mean of the distribution of θ.

the largest hypercone) that can be projected from the current point of interest (cone *vertex*), out between the other valid points. The centre vector (*axis*) of the hypercone is an approximation of the vector $\vec{n}$ in (1) and the directionality of the axis can be used to assess how relevant the point under consideration is (for minimisation/ maximisation). The nearer to the objective boundary that the point of interest lies, the greater the apex angle of the largest empty hypercone becomes, i.e. a surrounded point can only fit a very narrow cone between the other points identified in the objective region.

The definition in (1) does not provide an implicit description of an optimal solution, i.e. an algorithm based on (1) would both minimise and maximise all objectives in one run of the algorithm! It is sensible in practice to restrict the direction of vector $\vec{n}$ in (1) to always point with a component in the direction of an ideal utopia solution, for example if all objectives are to be minimised, then it would be sensible to constrain all elements of the vector $\vec{n}$ to be negative, pointing to the 'lower left' corner of the optimisation hyperspace. If some objectives are to be maximised, the preferred search direction can easily be constrained accordingly.

Unfortunately finding the largest projected hypercone through a set of N-dimensional points is itself not trivial and the calculation of an exact solution will rival the processing required for the Delaunay triangulation of the set of points. Alternatively the identification of the largest cone angle could be treated as an optimisation problem.

If a random unit-length vector $\vec{m}$ is chosen and projected from a point within the objective space that is being assessed, then the closest point in angle from the remainder of the set of objective points would form one vector which tracks the edge of an empty hypercone (a generatrix of the hypercone). If a random search is conducted of cone axis vectors $\vec{m}$, eventually a vector $\vec{m}$ would be identified which is the axis of the largest empty hypercone. Given that the probability of a random vector landing in a hypercone is highest for the hypercone of interest (i.e. by definition the largest empty hypercone has the largest solid angle), a simple random search could be employed.

In a 2D problem, if we consider a set of a large number of points distributed uniformly along a circular trajectory with a point-of-interest, P, at the cluster centre, we may analyse the behaviour of the random search theoretically. Figure 1b shows a typical configuration. If the distribution of the points are uniform along the circular trajectory, then the arc-length between neighbouring points, and therefore the cone angles between them will be distributed following an exponential distribution. Moreover, if we place a test-point at a location on the circle selected uniformly at random, the segment length and therefore the angle to the nearest neighbour solution will also be distributed exponentially [7].

The exponential behaviour for nearest neighbour distances does however require two assumptions to be satisfied:

1. The point density is sufficiently high so that the probability of observing the upper limit on the angle $\theta = 2\pi$ is very small,
2. the distribution of the points on the circle is from a uniform distribution.

The exponential distribution of the nearest neighbours is described by

$$f(x) = \lambda e^{-\lambda x} \tag{2}$$

$$F(x) = 1 - e^{-\lambda x}\,, \quad x \geq 0; \tag{3}$$

where $f(x)$ is the probability density function with mean $1/\lambda$ and F(x) is the cumulative distribution function that describes the probability of an observation from the distribution being less than or equal to x. For the nearest neighbour distribution the angle between neighbours will be distributed with $\lambda_\theta = N_l/2\pi$, giving a mean angle of $\bar{\theta} = 2\pi/N_l$; where N_l is the number of points on the circular trajectory ($N_l = 14$ in Fig. 1b). For the angle α between a random test vector $\vec{m}$ (distributed uniformly within the circle) and its nearest neighbour, the distribution of α will also be exponential, but with a mean that is half of $\bar{\theta}$; i.e. $\lambda_\alpha = N_l/\pi$. Although the theoretical analysis is for points lying on a circle, as a uniform distribution of points on the surface of a hypersphere may be generated by creating a vector with each axis drawn from the normal distribution N(0,1) and then normalised to unit length [8], the theory will also hold for the *angles* between a normally distributed cluster of points too, and in practice the angles between points uniformly distributed in a cartesian region [9].

In practice, a cluster of 7+ points will provide a practical lower limit to producing the anticipated exponential behaviour (assumption 1) as there is less than a 1 in 1000 chance of the exponential distribution ideally producing values greater than the maximum $\theta = 2\pi$ limit (calculated from the cumulative distribution in (3)). If a Gaussian mutation scheme is used within an evolutionary algorithm, often the localised distribution of neighbouring solutions is sufficiently uniform to satisfy assumption 2.

If we perform a random search by generating a series of test vectors, $\vec{m}$ and then taking the maximum nearest neighbour angle, then we can determine the resultant probability density distribution of the maximum observed angle by transforming (2) and (3) with the order-statistic formula [10]

$$f_{N_s,z}(x) = \frac{N_s!}{(z-1)!(N_s-z)!} F(x)^{z-1}(1-F(x))^{N_s-z} f(x); \tag{4}$$

where N_s is the number of random samples taken and z is the order statistic of interest. In this work we are interested in the maximum value and therefore $z = N_s$. Combining (2) and (3) with (4) yields (5)

$$f_{N_s}(x) = N_s \lambda \sum_{k=0}^{N_s-1} \frac{(N_s-1)!}{k!(N_s-1-k)!} (-1)^k e^{-\lambda(k+1)x} \tag{5}$$

$$F_{N_s}(x) = N_s \sum_{k=0}^{N_s-1} \frac{(N_s-1)!}{k!(N_s-1-k)!} \frac{(-1)^k}{k+1} \left(1 - e^{-\lambda(k+1)x}\right) \tag{6}$$

where $f_{N_s}(x)$ is the probability density distribution formed from taking the maximum of N_s angles and $F_{N_s}(x)$ is the cumulative probability distribution.

Thus a point can be assessed as being *interior* by:

1. Generating a random direction vector $\vec{m}$ and identifying the nearest neighbour in angle space. The nearest neighbour is found by taking the dot/inner product between $\vec{m}$ and the difference vector between the point under consideration, P and a set of N_l points $\{Q_j : j \in [1, N_l]\}$ which form a local neighbourhood around the point P. The dot product (if $\vec{m}$ and the difference vector are unit length) yields the cosine of the angle between the vectors. The smallest angle (largest cosine) is the nearest neighbour;
2. The process repeated for a different random unit length vector $\vec{m}$ and the largest overall observed neighbour angle (smallest cosine) and corresponding vector $\vec{m}$ are recorded.

The method is described mathematically as

$$\alpha = \cos^{-1}\left(\min_{i=1}^{N_s}\left(\max_{j=1}^{N_l}\left(\frac{\vec{m}_i \cdot (Q_j - P)}{|\vec{m}_i||(Q_j - P)|}\right)\right)\right). \tag{7}$$

The resultant angle α can then be considered as being associated with interior or exterior points by recognising that the cumulative probability distribution $F_{N_s}(\alpha)$ in (6) can be used to determine a threshold level for the observed angle. If the value of α needed to give $F_{N_s}(\alpha) = 0.95$ is found and used as a threshold, then there is a 95% probability that a point that is truly an interior point will be correctly classified as interior. The method can be verified experimentally and is accurate as long as the assumptions of sufficient point density and uniform distribution are observed (software that performs the experimental verification of the CDF is available at [11]).

Unfortunately the ideal case of being able to assess explicitly whether a point is exterior is not practical, as the probability density distribution of exterior points will vary depending on the degree of convexity of the objective front local to the point and is unknown and not trivial to observe. Practically, the threshold that can be calculated for interior point acceptance is sufficient to create a useful algorithm. If a higher threshold level (e.g. 99%) is set, the algorithm is more aggressive at classifying points as interior.

4 Extension to Many-Objectives

As the dimensionality of the objective space is increased, the apex angles of the largest empty hypercones that can be fitted between a set of points also tend to increase. Thus instead of specifying the limit on the smallest apex-angle before solution cropping, an alternative strategy has been employed that allows the apex angle to be normalised into a common space, removing any issues associated with objective space dimensionality.

If a hypercone is projected through a hypersphere, the region of the hypersphere surface that is contained within the cone is a hyperspherical cap. The normalisation is obtained by transforming the observed cone angles into the ratio of the area of the hyperspherical cap to the total hypersphere area. The key observation of this transformation is that even in very high dimensional spaces, the distribution of the hyperarea ratios between nearest neighbours still follows the exponential distribution theory for nearest neighbours that was observed in the two-dimensional case. For example, in two dimensions, in Fig. 1b, if the circle is unit radius, the arc length between the two neighbours is the hyperspherical cap and d_1 may be transformed into a ratio $R = d_1/2\pi$.

Equations 8 and 9 provide the description of the hypersphere and hyperspherical cap areas [12] for a hypersphere of radius r, and (12) describes the ratio of the areas, based on the axis-to-generatrix cone angle α as shown in fig. 1b; where α would be used to calculate the ratio of the arc d_2 to the total circumference. It is clear that to calculate (12), the processing requirement is linear in the number of objectives, n.

$$A_s = \begin{cases} \frac{r^{n-1} n \pi^{n/2}}{(n/2)!} & \text{n is even,} \\ \frac{r^{n-1} n \pi^{(n-1)/2} 2^{n+1} (\frac{n+1}{2})!}{(n+1)!} & \text{n is odd,} \end{cases} \tag{8}$$

$$A_c = \begin{cases} \frac{r^{n-1} n \pi^{(n-2)/2}}{(n/2)!} p & \text{n is even,} \\ \frac{r^{n-1} n \pi^{(n-1)/2} 2^{n} (\frac{n+1}{2})!}{(n+1)!} q & \text{n is odd,} \end{cases} \tag{9}$$

$$p = \begin{cases} \alpha & n = 2, \\ \alpha - \cos(\alpha) \sum_{j=0}^{\frac{n-4}{2}} \frac{2^{2j}(j!)^2}{(2j+1)!} \sin(\alpha)^{2j+1} & n \geq 4 \end{cases} \tag{10}$$

$$q = 1 - \cos(\alpha) \sum_{j=0}^{\frac{n-3}{2}} \frac{(2j)!}{2^{2j}(j!)^2} \sin(\alpha)^{2j} \tag{11}$$

$$R = \frac{A_c}{A_s} = \begin{cases} \frac{1}{\pi} p & \text{n is even,} \\ \frac{1}{2} q & \text{n is odd,} \end{cases} \tag{12}$$

The largest angle observed by the random sampling process can be converted to a ratio R using (12) and then the ratio compared against the limit calculated from (6) (with x representing ratio not angles) to test whether the point is interior or not. The configuration for many-objectives is the same as for the 2D case with $\lambda = N_l$ forming the shape parameter of the exponential distribution and N_s being the number of random samples used to estimate the largest cone angle (and therefore ratio). In practice, conditions of $N_s/N_l < 1/2$ and $N_s > 7$ are sufficient to ensure that the assumption that the distribution is exponential is valid. A local neighbourhood is best formed by selecting the N_l nearest neighbours using Euclidian distance in the objective space. If the local neighbourhood is small (e.g. in range 25 to 100) then the assumption that the points used for comparison follow a uniform distribution in the single dimension *ratio* space is also usually satisfied. The ratio R can also be used in the fitness assignment process to determine solution quality, with points having larger ratios being more 'interesting'.

5 Solution Spreading and Front Maintenance

As with most multi/many objective optimisation algorithms, some means of spreading the solutions evenly across the objective boundary is needed. Naturally, the solutions will tend to prefer convex regions if the ratio R alone is used as a selection criteria. For demonstration purposes with the proof-of-principle algorithm, a simple sharing mechanism has been employed. The sharing mechanism is to calculate the total weighted Euclidean distance to the N_l neighbouring solutions. The Euclidean distance to the N_l neighbours is needed in order to normalise the difference vectors for angle and ratio calculations, therefore the sharing calculation overhead is little more.

The sharing function is defined in (13), where $F_s(i)$ is the shared fitness of solution i, T is the set of N_l local solutions (including solution P), R_i is the transformed ratio for solution i, σ_s is the standard deviation of the sharing function and d_{ij} is the Euclidian distance between solutions i and j. As solution i will be compared with itself, the denominator of (13) will always be at least unity.

$$F_s(i) = \frac{R_i}{\sum_{j \in T} \exp(-\frac{d_{ij}^2}{2\sigma_s^2})} \tag{13}$$

The distance d_{ij} may be calculated in either objective or decision space, however for the proof-of-principle algorithm, only objective space sharing has been applied. For sensible sharing distances to be generated, it is advised to scale each of the objectives appropriately; in real engineering problems, this scaling information is often available.

All of the solutions that have been generated by the evolutionary algorithm so far are sorted so that the feasible solutions with the highest shared fitness are at the start of the list, and then any constrained solutions are appended after the list of sorted feasible solutions. The constrained solutions are sorted based upon their worst degree of constraint violation, i.e. each solution has a vector of constraint violations, with negative values indicating a violated constraint. The more negative the violations, the further the solution is from the constraint boundary; the minimum value of the constraint vector is used for solution sorting.

6 Evolutionary Algorithm Structure

A mutation only, evolutionary process has been employed, with an incremental structure. Each generation, the 100 top solutions are chosen from the sorted list of all solutions evaluated. These solutions are mutated with a normal distribution in each dimension. The standard deviation of the mutation is initially $\frac{1}{8}^{th}$ of the range of each decision variable, and the standard deviation is reduced by 10% each generation (i.e. multiplied by a factor of 0.9). To estimate the hypersphere ratio, a random search size of $N_s = 10$, local sample size of $N_l = 25$ and $\sigma_s = 0.001$ have been applied.

To reduce the computational effort, the hypercap ratios of most of the interior and a proportion of the crowded solutions are not evaluated each generation. Points that are considered as interior are not removed completely; they are marked as constrained but given a 50% probability of being selected for ratio re-calculation on the next iteration. If the point is still considered interior then its probability of re-calculation is reduced by a factor of 50% to 25% etc. Heavily crowded points are also marked as constrained but given a probability of 50% of being reconsidered on the next algorithm iteration.

By maintaining all solutions generated so far as possible candidates for future mutations, the algorithm does slow with increasing numbers of generations. However by not forgetting where previous solutions have been generated, the algorithm builds a good approximation to the objective front in few generations. The algorithm is best suited to initial coarse exploration of unknown problems to identify the underlying objective space structure and identify regions which warrant more focussed investigation with Many-Objective algorithms such as MSOPS-II [2].

7 Example Behaviour Results

As a demonstration of the behaviour of the algorithm, the Tanaka objective function has been used (detailed in [2]) but with an added constrained region of a small circle centred at (0.3 0.2) in the plots. Two runs, both for 50 generations have been conducted. The first run has no restriction on the direction of the boundary search and has identified the entire objective boundary, while for the second run, a preferred direction corresponding to minimisation has been imposed by restricting the components of $\vec{m}$ in (7) to be

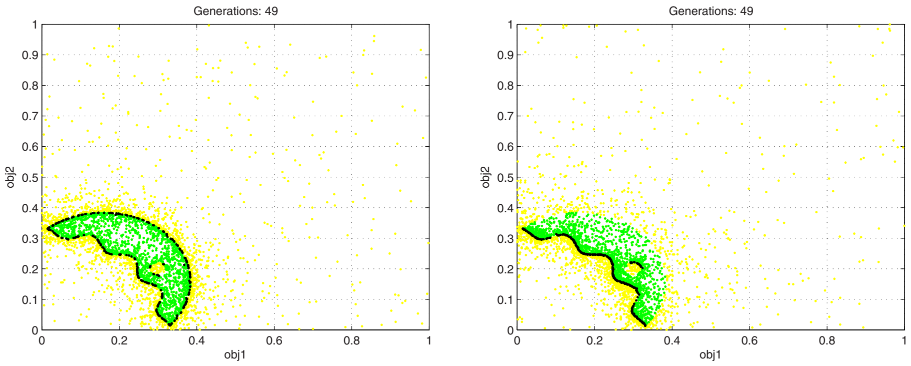


Fig. 2. Output of the prototype DOBI algorithm for the modified Tanaka function. The left plot shows the behaviour when the boundary search is unrestricted and the entire objective boundary is identified. The right figure shows the output when a restriction of minimisation is placed on the algorithm. In both plots, the convex/ concave regions of the objective front and the hole in the feasible region have been identified correctly. The light points are constrained solutions, mid grey are classed as interior and the black points are classed as exterior.

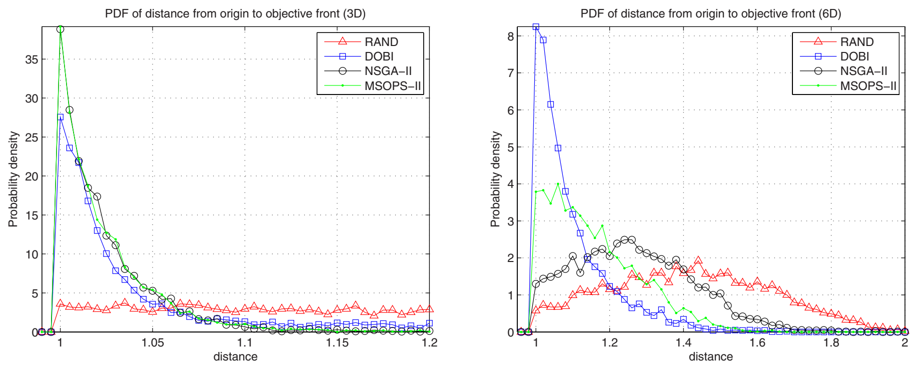


Fig. 3. Histograms of Pareto front distance for 5000 solution points from multiple runs on a unit hypersphere function of NSGA-II, MSOPS-II, DOBI (with restricted boundary search) and a random search . The left plot shows the behaviour for the 3-objective configuration; MSOPS-II and NSGA-II provide indistinguishable results with the DOBI algorithm a close 3^{rd}. The right plot shows the performance with 6 objectives. The DOBI algorithm is clearly superior, with the performance of NSGA-II falling as anticipated.

negative. Figure 2 shows the locations of the evaluated and classified solutions for the two conditions.

Figure 3 shows a comparison of the DOBI algorithm to other state-of-the-art methods. The test function has a Pareto front formed by a constraint boundary comprising the unit hypershpere (and hence is concave). The distance from the origin to identified objective vectors have been grouped to form histograms; an ideal case would have all solutions with a distance of unity. The results show clearly that the DOBI algorithm can be competitive in low dimensions and is superior for many-objective problems.

Rigourous statistical verification of the prototype algorithm behaviour is at present on-going; there are no existing metrics that allow the performance of the algorithm to be examined, and no other algorithms are capable of developing the entire objective boundary. For example the closest contender MSOPS-II can only form a limited objective front; it is not capable of identifying interior regions without a-priori knowledge of their location.

8 Conclusions

This paper has introduced a new and unique concept for an algorithm that is capable of identifying the entire boundary of the feasible objective space, even with internal holes, in many objective dimensions. The theory behind the exponential distribution of hyperspherical cap ratios has been introduced and used to construct a prototype evolutionary algorithm capable of unparalleled exploration behaviour.

By using a small random search and a normalisation into hyperspherical cap ratio space within the evolutionary algorithm, the time complexity of the search scales linearly with the number of objective dimensions, unlike Delaunay-based methods which become impractical for even moderate dimensionality. Prototype algorithm software that will re-produce all of the results in this paper is available for academic use at [11].

References

1. Hughes, E.J.: Radar waveform optimisation as a many-objective application benchmark. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 700–714. Springer, Heidelberg (2007)
2. Hughes, E.J.: MSOPS-II: A general-purpose many-objective optimiser. In: Congress on Evolutionary Computation CEC 2007, Singapore, pp. 3944–3951. IEEE, Los Alamitos (2007)
3. Purshouse, R.C.: On the Evolutionary Optimisation of Many Objectives. PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK (September 2003)
4. Hughes, E.J.: Evolutionary many-objective optimisation: Many once or one many? In: IEEE Congress on Evolutionary Computation, vol. 1, pp. 222–227. IEEE, Los Alamitos (2005)
5. Knowles, C., Deb, K. (eds.): Multiobjective Problem Solving from Nature: From Concepts to Applications. Natural Computing Series. Springer, Berlin (2008)
6. Zhao, H., Osher, S., Fedkiw, R.: Fast surface reconstruction using the level set method. In: Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision (VLSM 2001) (July 2001)
7. Jiřina, M.: Nearest neighbour distance statistics estimation. Technical report, Institute of Computer Science: Academy of Sciences of the Czech Republic (October 2002), `ftp://ftp.cs.cas.cz/pub/reports/v878-02.pdf`
8. Muller, M.E.: A note on a method for generating points uniformly on n-dimensional spheres. Communications of the ACM 2, 19–20 (1959)
9. Hicks, J.S., Wheeling, R.F.: An efficient method for generating uniformly distributed points on the surface of an n-dimensional sphere. Communications of the ACM 2, 17–19 (1959)
10. Rose, C., Smith, M.D.: Mathematical Statistics with Mathematica. Springer, Heidelberg (2002)
11. Hughes, E.J.: Software resources, `http://code.evanhughes.org`
12. Jacquelin, J.: Le problème de l'hyperchèvre. Quadrature (49), 6–12 (July- September 2003), `http://www.maths-express.com/articles/hyperchevre.pdf` ISSN 1142-2785

Use of Heuristic Local Search for Single-Objective Optimization in Multiobjective Memetic Algorithms

Hisao Ishibuchi, Yasuhiro Hitotsuyanagi, Noritaka Tsukamoto, and Yusuke Nojima

Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
{hisaoi@,hitotsu@ci.,nori@ci.,nojima@}cs.osakafu-u.ac.jp
http://www.ie.osakafu-u.ac.jp/~hisaoi/ci_lab_e

Abstract. This paper proposes an idea of using heuristic local search procedures specific for single-objective optimization in multiobjective genetic local search (MOGLS). A large number of local search techniques have been studied for various combinatorial optimization problems. Thus we may have a situation where a powerful local search procedure specific for a particular objective is available in multiobjective optimization. Such a local search procedure, however, can improve only a single objective. Moreover, it may have severe side-effects on the other objectives. For example, in a scheduling problem, an insertion move of a job with the maximum delay to an earlier position in a current schedule is likely to improve only the maximum tardiness. In this paper, we assume a situation where each objective has its own heuristic local search procedure. First we explain our MOGLS algorithm, which is the hybridization of NSGA-II and weighted sum-based local search. Next we propose an idea of using heuristic local search procedures specific for single-objective optimization in MOGLS. Then we implement the proposed idea as a number of variants of MOGLS. These variants are different from each other in the choice of a heuristic local search procedure. We examine three schemes: random, probabilistic and deterministic. Finally we examine the performance of each variant through computational experiments on multiobjective 0/1 knapsack problems with two, three and four objectives. It is shown that the use of heuristic local search procedures and their appropriate choice improve the performance of MOGLS.

1 Introduction

It has been demonstrated in the literature [1], [5], [7], [10], [24] that the search ability of evolutionary optimization algorithms can be improved by the hybridization with local search. Such a hybrid algorithm has often been referred to as memetic algorithms [25]. Implementation of memetic algorithms has been discussed in detail for single-objective optimization [6], [20], [22], [23]. Self adaptation of local search strategies has also been discussed for single-objective optimization [20], [26], [27].

Whereas most memetic algorithms have been developed for single-objective optimization, real-world application tasks usually involve multiple objectives. It is well-recognized that evolutionary algorithms are suitable for multiobjective optimization because a number of non-dominated solutions can be obtained by their single run.

A number of multiobjective evolutionary algorithms (MOEAs) have been proposed and successfully applied to various application areas [2]. Limitations of their search ability, however, have also been reported. One limitation is poor scalability to many-objective problems. Since almost all solutions in the current population become non-dominated with each other under the presence of many objectives, the search ability of MOEAs is severely deteriorated by the increase in the number of objectives. It was demonstrated in [8], [17] that MOEAs were outperformed by multiple runs of single-objective evolutionary algorithms (SOEAs). See [12], [13] for a review of various studies on evolutionary many-objective optimization. Another limitation is related to combinatorial optimization. It was demonstrated in [16] that MOEAs did not find a good non-dominated solution set that well approximated the entire Pareto front of a large-scale two-objective knapsack problem.

Hybridization with local search is a promising approach for overcoming the above-mentioned limitations. The hybridization, however, is not straightforward in the case of multiobjective optimization. In single-objective memetic algorithms (SOMAs), the same objective function can be used for global search and local search. Thus the hybridization is straightforward in the design of SOMAs. This is not the case in the design of multiobjective memetic algorithms (MOMAs) because local search is basically a single-objective optimization technique for finding a single optimal solution (while global search in MOMAs is supposed to search for a large number of non-dominated solutions with respect to multiple objectives). Thus we need to implement a local search procedure that can handle multiple objectives in the design of MOMAs.

The first MOMA, which is called a multiobjective genetic local search (MOGLS) algorithm, was proposed in [9]. In MOGLS, a weighted sum fitness function is used in parent selection and local search while Pareto dominance is used for updating a secondary population. In order to search for a variety of non-dominated solutions along the Pareto front, the weight vector in the weighted sum fitness function is randomly updated whenever a pair of parent solutions is chosen for crossover. The same weight vector is used in local search for an offspring solution generated by genetic operations from the chosen parents. A variant of MOGLS with higher search ability was proposed in [15]. It was demonstrated by Jaszkiewicz [16] that his MOGLS outperformed other MOMAs (i.e., M-PAES [18], [19] and the original MOGLS [9]) and a well-known MOEA (i.e., SPEA [28]). M-PAES (memetic Pareto archived evolution strategy) is an MOMA where Pareto dominance is used to compare the current solution with its neighbor in local search. When they are non-dominated, they are compared using a crowding measure based on a grid-type partition of the objective space.

Whereas high search ability of MOGLS was reported for multiobjective traveling salesman problems [15] and multiobjective knapsack problems [16], it did not significantly outperform MOEAs in its application to multiobjective flowshop scheduling [9], [14]. This may be because local search did not use any problem-specific knowledge in multiobjective flowshop scheduling in [9], [14]. On the other hand, good results were reported for a bi-objective capacitated arc routing problem in [21] where constructive heuristics were used for generating good initial solutions in MOMAs.

In this paper, we assume a situation where each objective has its own powerful heuristic local search procedure. Then we propose an idea of using such a heuristic local search procedure for single-objective optimization in MOGLS.

2 Multiobjective Genetic Local Search

Let us consider the following k-objective maximization problem:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ f_2(\mathbf{x}),\ ...,\ f_k(\mathbf{x}))\ , \tag{1}$$

where $\mathbf{f}(\mathbf{x})$ is a k-dimensional objective vector, and $\mathbf{x}$ is a decision vector. MOEAs such as NSGA-II [3] and SPEA [28] were designed to search for a non-dominated solution set that well approximated the entire Pareto front (i.e., the projection of the Pareto optimal solution set onto the objective space). MOMAs such as MOGLS [9]. and M-PAES [18], [19] were also designed for the same task.

In this paper, we use a simple MOGLS (S-MOGLS [11]) algorithm to evaluate the effect of local search procedures specific for single-objective optimization. The generation update mechanism of S-MOGLS is illustrated in Fig. 1. From the current population, an offspring population is generated by genetic operations in the same manner as NSGA-II. Local search is applied to some offspring. An improved population consists of offspring improved by local search. Good individuals are selected from the current, offspring and improved populations to form the next population.

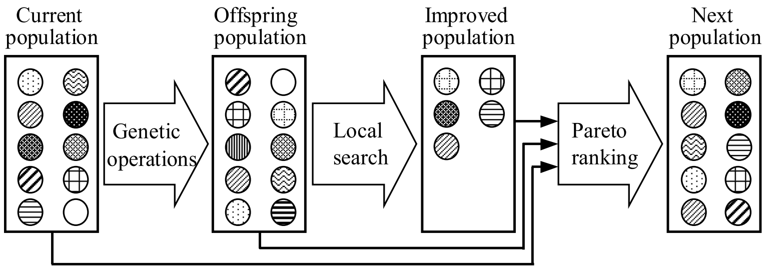


Fig. 1. Generation update in S-MOGLS

Let us denote the population size as N_{pop}. The size of the offspring population is the same as the current population. That is, N_{pop} offspring are generated by genetic operations. The size of the improved population depends on the number of offspring to which local search is applied. It also depends on the ability of local search to improve the current solution. The outline of S-MOGLS is written as follows:

[S-MOGLS]
```
Step 1: $P$ = Initialize($P$)
Step 2: While the stopping condition is not satisfied, do
Step 3:   $P'$ = Genetic Operations($P$)
Step 4:   $P''$ = Local Search($P'$)
Step 5:   $P$ = Generation Update($P \cup P' \cup P''$)
Step 6: End while
Step 7: Return Non-dominated($P$)
```

First an initial population with N_{pop} solutions is randomly generated in Step 1. Then Steps 3-5 are iterated until a prespecified stopping condition is satisfied. Finally non-dominated solutions in the final population are presented in Step 7.

Step 3 of S-MOGLS is exactly the same as NSGA-II. That is, each solution in the current population is evaluated by Pareto sorting and a crowding distance for parent selection. Then crossover and mutation are applied to a pair of selected parents. Step 5 is conceptually the same as NSGA-II. That is, the same fitness evaluation scheme as in NSGA-II is used to evaluate each solution in the current, offspring and improved populations. The best N_{pop} solutions are selected to form the next population.

In Step 4, we use the following weighted sum fitness function for local search:

$$f(\mathbf{x}) = \lambda_1 f_1(\mathbf{x}) + \lambda_2 f_2(\mathbf{x}) + \cdots + \lambda_k f_k(\mathbf{x}), \tag{2}$$

where $\boldsymbol{\lambda}=(\lambda_1, \lambda_2, ..., \lambda_k)$ is a weight vector. Before the execution of S-MOGLS, we first generate a set of uniformly distributed weight vectors. More specifically, we generate all integer vectors satisfying the following conditions:

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k = d \quad \text{and} \quad \lambda_i \in \{0, 1, ..., d\} \text{ for } i = 1,2,...,k . \tag{3}$$

For example, we generate six weight vectors (2, 0, 0), (1, 1, 0), (1, 0, 1), (0, 2, 0), (0, 1, 1), (0, 0, 2) when d is specified as $d = 2$ for three-objective problems (i.e., $k = 3$). In our computational experiments, d was specified as $d = 100$ for $k = 2$ (101 weight vectors). $d = 13$ for $k = 3$ (105 weight vectors), and $d = 7$ for $k = 4$ (120 weight vectors).

In Step 4, first a weight vector is randomly drawn from the weight vector set. Then an initial solution for local search is selected from the offspring population P' using tournament selection with replacement. Each offspring in P' is evaluated by the weighted sum fitness function in (2) with the current weight vector. In order to choose a good initial solution for local search, we use a large tournament size (e.g., 20 in our computational experiments). Local search is applied to the chosen initial solution with the local search application probability P_{LS}. The weighted sum fitness function in (2) with the current weight vector is used in local search.

In local search, a neighbor is randomly generated from the current solution. When a better neighbor is found, the current solution is replaced with the neighbor. That is, we use the first move strategy where local search accepts the first improved neighbor instead of the best improved neighbor in the neighborhood.

When a better neighbor is not found among a prespecified number of randomly drawn neighbors (say, L_{fail} neighbors), local search is terminated. That is, L_{fail} is the upper bound on the number of successive failure attempts. Local search is also terminated by the total number of examined neighbors (say, L_{search} neighbors) in a series of local search from a single initial solution. Local search is terminated when at least one of these two conditions is satisfied. In our computational experiments, L_{fail} and L_{search} were specified as $L_{\text{fail}} = 5$ and $L_{\text{search}} = 20$. If the initial solution is improved by local search, the final solution is added to the improved population P''. The selection of an initial solution and the application of local search with the local search application probability P_{LS} are iterated N_{pop} times where N_{pop} is the population size.

3 Use of Single-Objective Local Search in MOGLS

Let us assume that we have a general local search procedure $\mathrm{LS_G}$ which is not specific for any objective. We also assume that we have a set of k local search procedures $\mathrm{LS_S} = \{\mathrm{LS_{S1}}, \mathrm{LS_{S2}}, ..., \mathrm{LS_{S\mathit{k}}}\}$ where $\mathrm{LS_{S\mathit{i}}}$ is specific for the ith objective. Our problem in this section is how to choose one of the $(k+1)$ local search procedures for each solution (i.e., how to coordinate the local search procedures in local search).

Let us denote the selection probability of each local search procedure as P_G, P_S1, P_S2, ..., $P_{\mathrm{S}k}$ where $P_\mathrm{G}+P_\mathrm{S1}+P_\mathrm{S2}+ \ ... \ +P_{\mathrm{S}k}=1$. We also denote the selection probability of the specific local search procedures as P_S where $P_\mathrm{S}=P_\mathrm{S1}+P_\mathrm{S2}+ \ ... \ +P_{\mathrm{S}k}$.

In this paper, we handle P_G and P_S as user-definable parameters ($P_\mathrm{G}+P_\mathrm{S}=1$). That is, the values of P_G and P_S are prespecified. The specific local search procedures are used with the probability P_S while the general local search procedure $\mathrm{LS_G}$ is used with the probability P_G ($P_\mathrm{G}=1-P_\mathrm{S}$). Let us discuss how to choose one of the k specific local search procedures for each solution in local search. Since each specific local search procedure can improve only a single objective, it may be a good idea to use each of them with the same probability. A simple implementation of this policy is random choice. Another idea is to bias the selection probability of each specific local search procedure using the weight of the corresponding objective in the weighted sum fitness function. That is, we can use roulette wheel selection where the selection probability of each specific local search procedure is proportional to the weight of the corresponding objective. It is also possible to deterministically choose the specific local search procedure corresponding to the maximum weight. The above-mentioned three variants are summarized as follows:

Rand version: Random selection of one of the k specific local search procedures.
Prob version: Probabilistic selection where $P_\mathrm{S1}:P_\mathrm{S2}: \ ... :P_{\mathrm{S}k} = \lambda_1 : \lambda_2 : \ ... \ : \lambda_k$.
Max version: Deterministic selection using the maximum weight.

We also examine two variants with respect to the timing of the choice (i.e., the timing of the change) of a local search procedure. In a short term variant, the selected local search procedure is used for generating only a single neighbor. A different local search procedure is selected to generate another neighbor. In a long term variant, a local search procedure continues to be used in a series of local search from a single initial solution until local search is terminated. That is, the selection of a local search procedure is performed only when a new initial solution is chosen for local search. These two variants are denoted as follows:

Short version: A local search procedure is chosen whenever a neighbor is to be generated in local search. The chosen local search procedure is used for generating only a single neighbor.
Long version: A local search procedure is chosen only when an initial solution is selected for local search.

We examine all the 3×2 combinations (i.e., six variants). These six variants are referred to as MOGLS-GS because they use both general and specific local search.

4 Computational Experiments

In our computational experiments, we used nine multiobjective 0/1 knapsack problems in Zitzler & Thiele [28]. Their multiobjective 0/1 knapsack problems with k knapsacks (i.e., k objectives and k constraints) and n items can be written as follows:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})), \tag{4}$$

$$\text{subject to } \sum_{j=1}^{n} w_{ij} x_j \le c_i \;, \quad i = 1, 2, ..., k \;, \tag{5}$$

$$\text{where } f_i(\mathbf{x}) = \sum_{j=1}^{n} p_{ij} x_j \;, \quad i = 1, 2, ..., k \;. \tag{6}$$

In this formulation, $\mathbf{x}$ is an n-dimensional binary vector, p_{ij} is the profit of item j according to knapsack i, w_{ij} is the weight of item j according to knapsack i, and c_i is the capacity of knapsack i. Each solution $\mathbf{x}$ is handled as a binary string of length n. We denote the k-objective n-item knapsack problem as a k-n problem.

The performance of several MOEAs was examined in [28] using nine problems with two, three, four objectives and 250, 500, 750 items. We used these nine problems while we can report a part of our experimental results due to the page limitation.

Zitzler & Thiele [28] used a greedy repair procedure where items were removed in the increasing order of the maximum profit/weight ratio q_j over all knapsacks:

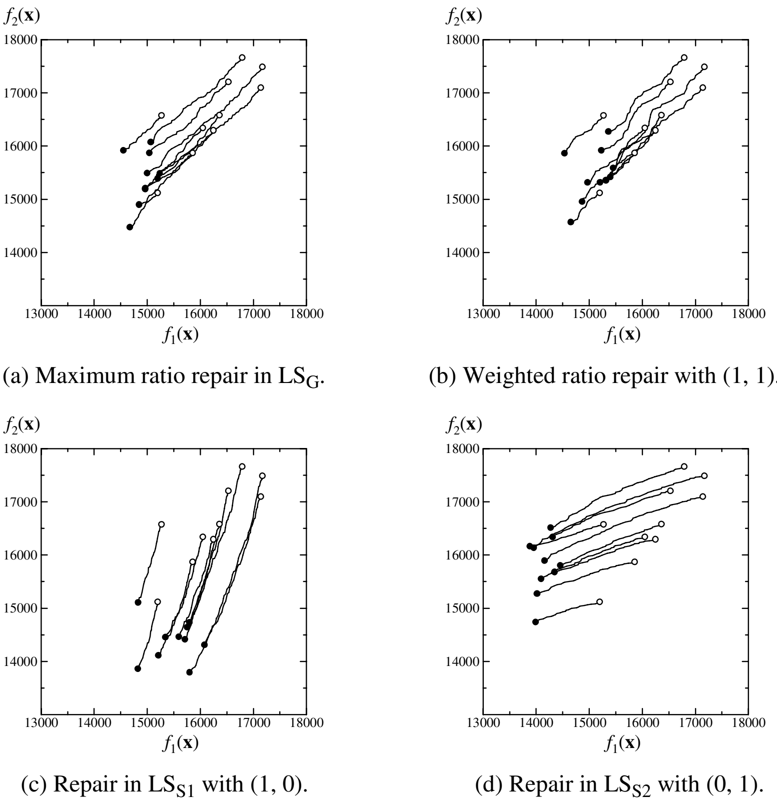
$$q_j = \max\{ p_{ij} / w_{ij} \mid i = 1, 2, ..., k \}, \; j = 1, 2, ..., n \;. \tag{7}$$

In our computational experiments, we used this maximum ratio repair in NSGA-II and S-MOGLS. On the other hand, Jaszkiewicz [16], [17] took into account the weight vector of the weighted sum fitness function in his greedy repair as follows:

$$q_j = \sum_{i=1}^{k} \lambda_i p_{ij} \Big/ \sum_{i=1}^{k} w_{ij} \;, \; j = 1, 2, ..., n \;. \tag{8}$$

In local search, we generated a neighbor in the following manner. First we applied the bit-flip operation to each bit of the current solution with the probability $4/n$ where n is the string length. Then we repaired the generated solution if it was not feasible. In S-MOGLS, we always used the maximum ratio repair in (7). On the other hand, the local search procedure with the maximum ratio repair was used as a general local search procedure $\mathrm{LS_G}$ in MOGLS-GS. As a specific local search procedure $\mathrm{LS}_{\mathrm{S}i}$, we used the local search procedure with the weighted ratio repair in (8) where only the weight value λ_i for the ith objective was 1 (all the other weight values were 0).

In Fig. 2, we show how infeasible solutions are repaired by each repair scheme. Randomly generated infeasible solutions (open circles) are repaired to feasible solutions (closed circles) in each plot in Fig. 2. We can see that similar results are obtained by the maximum ratio repair in Fig. 2 (a) and the weighted ratio repair with the weight vector (1, 1) in Fig. 2 (b). That is, the maximum ratio repair used in $\mathrm{LS_G}$ is not specific for any objective. On the other hand, feasible solutions with good values of



(a) Maximum ratio repair in LS_G. (b) Weighted ratio repair with (1, 1).

(c) Repair in LS_{S1} with (1, 0). (d) Repair in LS_{S2} with (0, 1).

Fig. 2. Effects of different repair schemes

the first objective were obtained by the greedy repair used in LS_{S1} in Fig. 2 (c). This observation shows that LS_{S1} can be viewed as a local search procedure specific for the first objective. The same observation can be obtained in Fig. 2 (d) for LS_{S2}.

We applied NSGA-II, S-MOGLS and MOGLS-GS to each test problem 50 times using the following parameter specifications: The population size was 200, the termination condition was the examination of 400,000 solutions, the crossover probability was 0.8 (uniform crossover), the mutation probability was $1/n$ (n: the string length), the local search application probability P_{LS} was 0.1, and the probability P_S of the specific local search procedures in MOGLS-GS was 0.5.

In Fig. 3, we show the 50% attainment surface [4] obtained by each algorithm for the 2-500 problem. As MOGLS-GS, we used the Long & Max version. From Fig. 3, we can see that much better results were obtained by MOGLS-GS than S-MOGLS with respect to the diversity of solutions along the Pareto front. This is because we used the specific local search procedure for each objective in MOGLS-GS.

In Tables 1-3, we show experimental results by MOGLS-GS with various parameter specifications. Each table shows the normalized average values of the hypervolume measure. All experimental results are normalized using the average result by

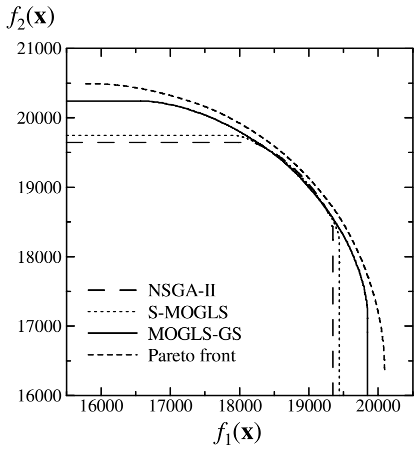


Fig. 3. The 50% attainment surface obtained by each algorithm for the 2-500 problem

NSGA-II. The best result in each table is highlighted by bold face print. From these tables, we can see that more frequent use of the specific local search procedures (i.e., the increase of P_S) improved the performance of MOGLS-GS. Better results were obtained from the long term version than the short term version. We can also see that the random choice of a specific local search procedure was outperformed by the probabilistic choice and the deterministic choice based on the weight vector.

Table 1. Average relative values of the hypervolume measure for the 2-500 problem

Versions of MOGLS-GS	NSGA-II	S-MOGLS	MOGLS-GS				
			P_S=0.1	P_S=0.2	P_S=0.3	P_S=0.4	P_S=0.5
Short & Rand	100.000	100.910	102.770	103.224	103.694	104.010	104.218
Short & Prob	100.000	100.910	103.305	103.929	104.313	104.584	104.887
Short & Max	100.000	100.910	103.352	103.999	104.441	104.717	104.927
Long & Rand	100.000	100.910	103.071	103.602	104.045	104.283	104.487
Long & Prob	100.000	100.910	103.589	104.257	104.446	104.762	104.997
Long & Max	100.000	100.910	103.688	104.250	104.631	104.873	**105.066**

Table 2. Average relative values of the hypervolume measure for the 3-500 problem

Versions of MOGLS-GS	NSGA-II	S-MOGLS	MOGLS-GS				
			P_S=0.1	P_S=0.2	P_S=0.3	P_S=0.4	P_S=0.5
Short & Rand	100.000	101.581	103.409	104.538	105.178	105.649	106.007
Short & Prob	100.000	101.581	104.710	106.275	107.183	108.124	108.662
Short & Max	100.000	101.581	105.025	106.740	107.753	108.557	109.141
Long & Rand	100.000	101.581	104.304	105.518	106.192	106.680	107.512
Long & Prob	100.000	101.581	105.686	107.387	108.203	108.833	109.278
Long & Max	100.000	101.581	106.380	107.813	108.606	109.225	**109.608**

Table 3. Average relative values of the hypervolume measure for the 4-500 problem

Versions of MOGLS-GS	NSGA-II	S-MOGLS	MOGLS-GS				
			P_S=0.1	P_S=0.2	P_S=0.3	P_S=0.4	P_S=0.5
Short & Rand	100.000	102.288	104.020	105.361	106.288	106.986	107.668
Short & Prob	100.000	102.288	106.151	107.880	109.387	110.210	110.663
Short & Max	100.000	102.288	106.667	108.836	110.031	110.941	112.108
Long & Rand	100.000	102.288	105.175	106.747	107.754	108.332	108.828
Long & Prob	100.000	102.288	107.623	109.030	110.584	111.109	111.876
Long & Max	100.000	102.288	108.154	110.291	111.419	111.959	**112.696**

5 Conclusions

We proposed an idea of using heuristic local search procedures specific for single objective optimization in multiobjective memetic algorithms (MOMAs). Through computational experiments on multiobjective 0/1 knapsack problems, we demonstrated that the proposed idea improved the performance of MOMAs. Whereas we assumed that all objectives had powerful local search procedures, this is not the case in real-world applications. An interesting future research issue is the handling of a more general situation where a heuristic local search procedure for each objective has different search ability: Some are very powerful while others are not (including a situation where only a part of the objectives have heuristic local search procedures).

This work was partially supported by Grant-in-Aid for Scientific Research (B): KAKENHI (20300084).

References

1. Bersini, H., Dorigo, M., Langerman, S., Seront, G., Gambardella, L.: Results of the First International Contest on Evolutionary Optimization. In: Proc. of 1996 IEEE International Conference on Evolutionary Computation, pp. 611–615 (1996)
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6, 182–197 (2002)
4. Fonseca, C.M., Fleming, P.J.: On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 584–593. Springer, Heidelberg (1996)
5. Freisleben, B., Merz, P.: A Genetic Local Search Algorithm for Solving Symmetric and Asymetric Traveling Salesman Problems. In: Proc. of 1996 IEEE International Conference on Evolutionary Computation, pp. 616–621 (1996)
6. Hart, W.E.: Adaptive Global Optimization with Local Search. Ph. D. Thesis, University of California, San Diego (1994)
7. Hart, W.E., Krasnogor, N., Smith, J.E. (eds.): Recent Advances in Memetic Algorithms, pp. 3–27. Springer, Berlin (2005)
8. Hughes, E.J.: Evolutionary Many-Objective Optimization: Many Once or One Many? In: Proc. of IEEE Congress on Evolutionary Computation, pp. 222–227 (2005)

9. Ishibuchi, H., Murata, T.: A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews 28, 392–403 (1998)
10. Ishibuchi, H., Murata, T., Tomioka, S.: Effectiveness of Genetic Local Search Algorithms. In: Proc. of 7th International Conference on Genetic Algorithms, pp. 505–512 (1997)
11. Ishibuchi, H., Narukawa, K.: Some Issues on the Implementation of Local Search in Evolutionary Multiobjective Optimization. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1246–1258. Springer, Heidelberg (2004)
12. Ishibuchi, H., Tsukamoto, N., Hitotsuyanagi, Y., Nojima, Y.: Effectiveness of Scalability Improvement Attempts on the Performance of NSGA-II for Many-Objective Problems. In: Proc. of Genetic and Evolutionary Computation Conference (in press, 2008)
13. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary Many-Objective Optimization: A Short Review. In: Proc. of 2008 Congress on Evolutionary Computation, pp. 2424–2431 (2008)
14. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling. IEEE Trans. on Evolutionary Computation 7, 204–223 (2003)
15. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization. European Journal of Operational Research 137, 50–71 (2002)
16. Jaszkiewicz, A.: On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem - A Comparative Experiment. IEEE Trans. on Evolutionary Computation 6, 402–412 (2002)
17. Jaszkiewicz, A.: On the Computational Efficiency of Multiple Objective Metaheuristics: The Knapsack Problem Case Study. European Journal of Operational Research 158, 418–433 (2004)
18. Knowles, J.D., Corne, D.W.: M-PAES: A Memetic Algorithm for Multiobjective Optimization. In: Proc. of 2000 IEEE Congress on Evolutionary Computation, pp. 325–332 (2000)
19. Knowles, J.D., Corne, D.W.: A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization. In: Proc. of 2000 Genetic and Evolutionary Computation Conference Workshop Program: WOMA I, pp. 103–108 (2000)
20. Krasnogor, N.: Studies on the Theory and Design Space of Memetic Algorithms. Ph. D. Thesis, University of the West of England, Bristol (2002)
21. Lacomme, P., Prins, C., Sevaux, M.: A Genetic Algorithm for a Bi-Objective Capacitated Arc Routing Problem. Computers & Operations Research 33, 3473–3493 (2006)
22. Land, M.W.S.: Evolutionary Algorithms with Local Search for Combinatorial Optimization. Ph. D. Thesis, University of California, San Diego (1998)
23. Merz, P.: Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscape and Effective Search Strategy. Ph. D. Thesis, University of Siegen, Siegen (2000)
24. Merz, P., Freisleben, B.: Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem. IEEE Trans. on Evolutionary Computation 4, 337–352 (2000)
25. Moscato, P.: Memetic Algorithms: A Short Introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 219–234. McGraw-Hill, London (1999)
26. Ong, Y.S., Keane, A.J.: Meta-Lamarckian Learning in Memetic Algorithms. IEEE Trans. on Evolutionary Computation 8, 99–110 (2004)
27. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of Adaptive Memetic Algorithms: A Comparative Study. IEEE Trans. on Systems, Man, and Cybernetics: Part B - Cybernetics 36, 141–152 (2006)
28. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans. on Evolutionary Computation 3, 257–271 (1999)

Distance Based Ranking in Many-Objective Particle Swarm Optimization

Sanaz Mostaghim and Hartmut Schmeck

Institute AIFB, University of Karlsruhe, Germany
{sanaz.mostaghim,hartmut.schmeck}@kit.edu

Abstract. Optimization problems with many objectives open new issues for multi-objective optimization algorithms and particularly Particle Swarm Optimization. Many of the existing algorithms are able to solve problems of low number of objectives, but as soon as we increase the number of objectives, their performances get even worse than random search methods. This paper gives an overview on Multi-objective Particle Swarm Optimization when having many objectives and parameters. Furthermore, two new variants of MOPSO are proposed which are based on ranking of the non-dominated solutions. The proposed distance based ranking in MOPSO improves the quality of the solutions for even very large objective and parameter spaces. The quality of the new proposed MOPSO methods has been tested and compared to the random search and NSGA-II methods. The tests cover 3 to 20 objectives and 20 to 100 parameters.

1 Introduction

Multi-Objective Particle Swarm Optimization (MOPSO) has been introduced to solve various optimization problems from science and industry in the last years [13,11]. The main features of MOPSO are the simplicity in the implementation and the small number of control parameters comparing to the Evolutionary Multi-objective Optimization (EMO) algorithms. In MOPSO, the positions of the candidate solutions (particles) in the parameter space are updated through the generations where in EMO, only the good solutions have the chance to survive the selection and recombination process [6,4].

MOPSO and EMO have different natures; it can be observed that the EMO methods have very well-distributed populations during the generations where MOPSO particles are moving in a swarm all close to each other. Besides the differences, both MOPSO and EMO methods evaluate the quality of the solutions based on the domination criteria. The non-dominated solutions are considered as the best solutions of a population. In MOPSO, the global best particles are selected using the non-dominated set [21,16,18] and they lead the population to move toward the optimal front. However, in a large objective space, since all the population members get non-dominated, there is a difficulty for MOPSO to efficiently move the particles. There might be a small move because of the randomness in the diversity preserving method or the boundary handling.

In EMO, this feature has been already studied [20,19,2,5,14], while in the literature of MOPSO there is no work on this aspect to the knowledge of the authors. In a recent work [5], Corne and Knowles studied some existing methods such as weighted average ranking method [2], K-Optimality [9], and the favour relation [10] in EMO and apply them to a TSP problem with 5 to 20 objectives. They conclude that the ranking method has the best performance among the others for the specific test problem.

In this paper, we study MOPSO methods in high dimensional parameter and objective space and propose two new variants of MOPSO which are based on the ranking proposed by [2] and a new distance based ranking method. The new distance based ranking method considers the positions of the non-dominated solutions which are located in less crowded areas and assigns them better ranks than the solutions in crowded areas. These ranking mechanisms in MOPSO are used for selecting the global best particles. The non-dominated solutions with good ranks have a higher chance to be selected. For this purpose two selection mechanisms such as fitness proportional and tournament selection are used. Furthermore, we analyze the performance of the MOPSO variants in large parameter and objective spaces and compare them with a random search method as a baseline and NSGA-II [7]. The results show that the performance of MOPSO is improved when using the distance based ranking method comparing with the results of a typical MOPSO and the other methods on the selected test problem.

This paper is structured as follows. Next section is a short description of multi-objective optimization and the ranking of the non-dominated solutions. Section 3 is about a new distance based ranking method which is used in Section 4. In Section 4, we briefly explain the MOPSO methods and the new distance based ranking mechanism in MOPSO. Experiments and results are shown in Section 5 and the paper is concluded in the last section.

2 Multi-objective Optimization

A multi-objective optimization problem is of the form

$$\text{minimize} \quad f = \{f_1(x), f_2(x), \cdots, f_m(x)\} \tag{1}$$

subject to $x \in S$, involving $m(\geq 2)$ conflicting objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$ that we want to optimize simultaneously. The parameters $x = (x_1, x_2, \cdots, x_n)^T$ belong to the feasible region $S \subset \mathbb{R}^n$. We denote the image of the feasible region by $Z \subset \mathbb{R}^m$ and call it a feasible objective region. The elements of Z are called objective vectors and they consist of objective values $f(x) = (f_1(x), f_2(x), \cdots, f_m(x))$.

A parameter vector $x_1 \in S$ is said to *dominate* a parameter vector $x_2 \in S$, iff (a) x_1 is not worse than x_2 in all objectives and (b) x_1 is strictly better than x_2 in at least one objective. $x_1 \in S$ is called *Pareto-optimal* if there does not exist another $x_2 \in S$ that dominates it. Finally, an objective vector is called Pareto-optimal if the corresponding decision vector is Pareto-optimal. The main goal of multi-objective optimization algorithms is to approximate the Pareto-optimal solutions by a set of well-distributed solutions.

2.1 Ranking Non-dominated Solutions

All the non-dominated solutions are said to be indifferent to each other. But if we have a set of N non-dominated solutions, we can rank them in terms of their positions, e.g., in crowded or uncrowded areas. Also there is a difference between two non-dominated solutions with respect to the number of objectives where one of them is

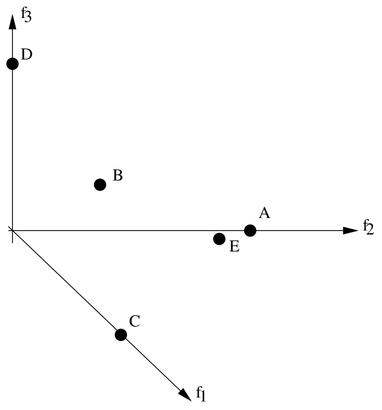


Fig. 1. An example showing 5 different non-dominated solutions in the objective space. The solutions A and E build a crowded area. D is the farthest solution to the crowded area and the solution B is located in the middle of the front.

better than the other. The so called weighted average ranking [2,5] assigns a rank to a non-dominated solution in terms of the number of objectives that are better than the other non-dominated solutions. Consider the two non-dominated solutions x_i and x_j. We compute a vector a_{ij} with m elements as follows:

$$a_{ijk} = \begin{cases} 1, \text{ if } f_k(x_i) < f_k(x_j) \\ 0, \text{ if } f_k(x_i) = f_k(x_j) \\ -1, \text{ if } f_k(x_i) > f_k(x_j) \end{cases} \tag{2}$$

where $k = 1, \cdots, m$ and the corresponding rank $AR(x_i)$ for the non-dominated solution x_i is:

$$AR(x_i) = \sum_k \sum_{j \neq i} \{(N+1) - a_{ijk}\} \tag{3}$$

The AR ranking mechanism considers the differences between the objective values in terms of being better or not. For each solution, we measure the number of objectives that are better than the other solutions and the rank is the sum of them. If a solution is better in most of the objectives, it will obtain a lower (better) rank.

Example. Consider the objective values of the 5 solutions $x_A \cdots x_E$ shown in Figure 1 with $f(x_A) = (0,10,0)$, $f(x_B) = (5,5,5)$, $f(x_C) = (10,0,0)$, $f(x_D) = (0,0,10)$, and $f(x_E) = (1,9,0)$. The AR values are $AR(x_A) = 71$, $AR(x_B) = 76$, $AR(x_C) = 71$, $AR(x_D) = 70$, and $AR(x_E) = 72$. In this example, the order of ranks from best to worst is D, A and C, E and B. D has the best rank because there are at least three other solutions, B, E and A which are worst in two objectives than B. In this ranking A and C have the same ranks which is not desirable. C must be preferred to A, since E is very close to A. □

3 Distance Based Ranking

The AR ranking method (described in the last section) does not consider the positions of the solutions and the distances between them in the objective space. For instance, the solutions A and C have the same ranks, but they have different locations and neighbors. A has a very close neighbor, E, whereas the closest neighbor to C is B.

In this section, we study a new variant of the ranking method by considering the distances between the objective vectors. We compute a new distance vector $d_{ij} = (|f_1(x_i) - f_1(x_j)|, \cdots, |f_k(x_i) - f_k(x_j)|)$ which indicates the absolute distance values between the objectives of the two solutions x_i and x_j.

We use the vector d_{ij} and compute a DR value (DR is very similar to AR but for simplicity the term $(N+1)$ is omitted):

$$DR(x_i) = \sum_k \sum_{j \neq i} d_{ijk} \tag{4}$$

Here, in the contrary to AR ranks, the solutions with high ranks indicate the preferred solutions which play a key role in keeping the information about the uncrowded areas in the objective space.

In the example from the last section, the DR values are $DR(x_A) = 57$, $DR(x_B) = 58$, $DR(x_C) = 73$, $DR(x_D) = 75$, and $DR(x_E) = 53$. The DR values rank the solutions in a totally different way than the AR values. In this ranking, the solutions from the best to worst are: D, C, B, A and E. D obtains the best rank (as in AR) but the next ranked solutions are C and B.

4 Multi-objective Particle Swarm Optimization

A typical MOPSO contains a population of particles which explore the parameter space by moving with particular velocities toward the optimal solutions. The velocity of each particle is influenced by a social impact coming from the population and the individual experience of the particle. In MOPSO, a set of particles may be considered as a population P_t in the generation t. Each particle i has a position defined by $x^i = \{x^i_1, x^i_2, \cdots, x^i_n\}$ and a velocity defined by $v^i = \{v^i_1, v^i_2, \cdots, v^i_n\}$ in the parameter space S. In generation $t+1$, the new velocity and position for each particle i is computed by:

$$\begin{aligned} v^i_{j,t+1} &= wv^i_{j,t} + c_1R_1(p^i_{j,t} - x^i_{j,t}) + c_2R_2(p^{i,g}_{j,t} - x^i_{j,t}) \\ x^i_{j,t+1} &= x^i_{j,t} + v^i_{j,t+1} \end{aligned} \tag{5}$$

where $j = 1, \cdots, n$, w is called the inertia weight, c_1 and c_2 are two positive constants, and R_1 and R_2 are random values in the range $[0,1]$.

In Equation (5) p^i_t is the best position that particle i could find so far (personal best particle). It is like a memory for the particle i which gets updated in each generation. One good strategy in selecting p^i_t is called the newest method [3]. This method compares the new position of the particle with p^i_t. If p^i_t is dominated by the new position or if they are indifferent to each other, p^i_t is replaced by the new position.

In Equation (5), $p_t^{i,g}$ is the position of the global best particle. Since in Multi-objective Optimization there is no single optimal solution, the global best particle is selected from the set of non-dominated solutions. In most of the MOPSO methods [21], the non-dominated solutions are stored in an archive and each particle selects its own global best from the archive. There are several strategies to find the global best and it has been shown that selecting the global best particles has a high impact on the diversity and convergence of the solutions. In the literature, Non-dominated sorting [15], Epsilon dominance [21], Pareto-dominance Concept [1] and the Sigma method [18] are some of the strategies for selecting the global best particles. The main criticism to all of these domination based methods is that as soon as we have a high dimensional objective space, all of the particles in the population get non-dominated and therefore the positions of the particles cannot be improved. In some cases, like in the Sigma method, it can be observed that the population stops moving [18]. The only existing MOPSO method (known to the authors) which is not based on the domination criteria is the Maximin fitness ranking in MOPSO [16]. This approach does not compute the non-dominated solutions at all. But it ranks all the particles in terms of their Maximin Fitness values. The small fitness values indicate the non-dominated solutions in a non-crowded area.

One good strategy for selecting the global best particles in high dimensional objective spaces is to randomly select the global best particles from the archive. In this way, we can force the population (of almost non-dominated solutions) to move.

Diversity preserving methods are also applied to MOPSO in order to avoid particles to converge to local optima. There are several approaches to prevent the premature stagnation of the basic PSO and MOPSO which introduce randomness into the swarm [17,22]. Most of the main approaches randomly reinitialize some solutions in each generation with a probability. This is called turbulence factor [12].

4.1 Ranking and Distance Based Ranking MOPSO

Here we propose a new strategy in selecting the global best particles in MOPSO. We compute the non-dominated solutions of the population in each generation and store them in an archive. For selecting the global best particle for each particle in the population, we perform the following steps. First, we apply one of the ranking methods (AR or the DR values) from Sections 2 and 3 to the non-dominated solutions and rank the solutions. We call the MOPSO using the AR ranking, RMOPSO and using the DR ranking DMOPSO. After the ranking, one of the non-dominated solutions is selected by a selection mechanism as the global best particle. The most common selection mechanisms are the fitness proportional and the tournament selection [6]. In the selection process, the solutions with high ranks must have a higher chance to be selected as the global best particles when using the DR ranking mechanism. For the AR ranking, the lowest ranked solutions are the best and the selection criterion is vice versa.

5 Experiments

The main goal of the experiments is to observe the behavior of MOPSO (a typical MOPSO and the new variants) in high dimensional parameter and objective spaces. We select one scalable test function from [8]. This test function can have n parameters and m objectives as follows:

$$\begin{aligned} f_1(x) &= (1+g(X_M))\cos(x_0\pi/2)\cdots\cos(x_{M-1}\pi/2) \\ f_2(x) &= (1+g(X_M))\cos(x_0\pi/2)\cdots\sin(x_{M-1}\pi/2) \\ &\vdots \\ f_{m-1}(x) &= (1+g(X_M))\cos(x_0\pi/2)\sin(x_1\pi/2) \\ f_m(x) &= (1+g(X_M))\sin(x_0\pi/2) \\ g(X_M) &= \sum_{j=M}^{n}(x_j-0.5)^2 \end{aligned} \tag{6}$$

where $x_i \in [0,\ 1]$, for $i = 0,\cdots,n$ and M is selected to be m. The Pareto-optimal solutions of this test function construct a hyper-sphere shape in the objective space. All the optimal solutions on the hyper-sphere have the property $S = \sum_{j=0}^{m} f_i^2(x) = 1$. We use the S values to evaluate the solutions in the high dimensional objective spaces. One can compute the number of solutions which fulfill the $S = 1$ criteria. The quality of different methods can be compared in terms of the number of solutions having S values close to one. In the experiments, this evaluation is done by computing the histograms of S for the solutions.

5.1 Parameter Setting

In the experiments, we vary the number of parameters and objectives from 20 to 100 and 3 to 20, respectively and find the optimal solutions of a typical MOPSO, ranking MOPSO (RMOPSO), Distance based ranking MOPSO (DMOPSO), Random Search, and NSGA-II [7].

We select a typical MOPSO as described in Section 4. The global best particles are selected at random from the archive. The personal best particles are selected based on the newest method. RMOPSO and DMOPSO indicate MOPSO methods which use the ranking and distance based ranking methods to rank the non-dominated solutions. The selection mechanisms in RMOPSO and DMOPSO are selected to be Fitness Proportional (FP) and Tournament selection with tournament size 5 indicated as TR(5). All the MOPSO methods are run with 100 particles for 100 generations and 100 non-dominated solutions are kept in an archive using a clustering technique [18]. The inertia weight is set to 0.4. The turbulence factor of 0.1 is used for all of the MOPSO variants and c_1 and c_2 are set to one.

The NSGA-II[1] method is run for the same number of evaluations, 100 individuals and 100 generations. The cross-over and mutation values are set to 0.9 and 0.2 as suggested in the program. The random search method selects 100 non-dominated solutions out of the non-dominated solutions from 10000 evaluations (the same number of evaluations used for all other methods). The selection is based on the clustering method used for other MOPSO methods. The reason for clustering is that the size of the non-dominated set drastically increases through the generations. All the programs are run for 20 different initial seeds. The average values are shown.

[1] The program is downloaded from www.iitk.ac.in/kangal/codes.shtml

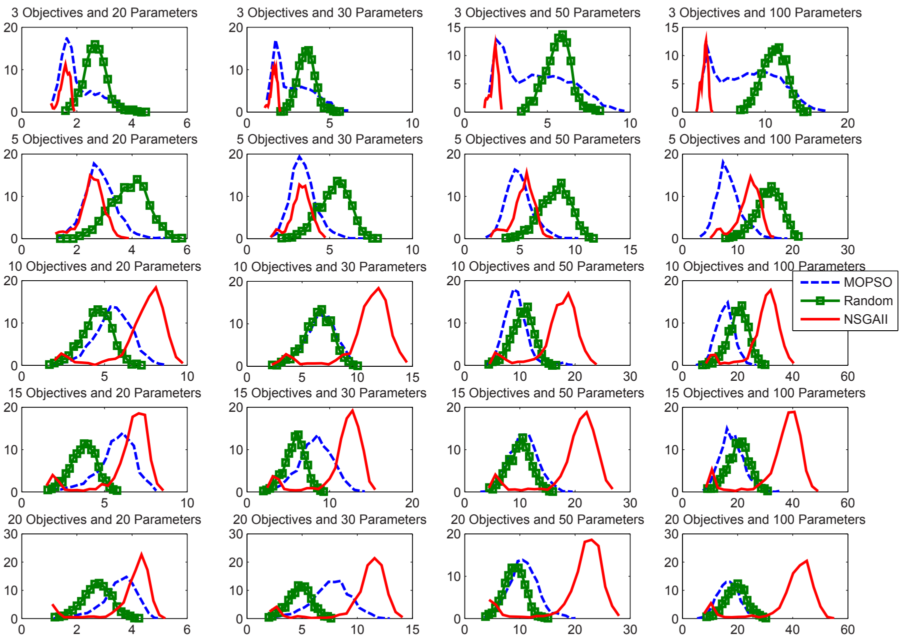


Fig. 2. Histogram of the S values for the Random, MOPSO and NSGA-II results for different numbers of parameters and objectives

5.2 Results

The first experiments are carried out to observe the quality of the solutions for different numbers of parameters and objectives for MOPSO, NSGA-II and the random method. Figure 2 shows the histograms of the S values for the different numbers of parameters and objectives. We observe that for the number of objectives less than 5, NSGA-II has the best performance among the others even if we increase the number of parameters to 100 (first row in Figure 2). For low number of parameters and the number of objectives more than 10, the random method is performing as the best and MOPSO has a better performance than NAGAII. For 20 objectives and 100 parameters, MOPSO and random have similar S values. This is due to the randomness in selecting the global best particles in MOPSO. Note that the optimal solutions must have $S = 1$ and none of our examined methods for $n > 50$ and $m > 10$ are able to find solutions with $S = 1$.

We furthermore analyze the influence of the ranking and the distance based ranking method on MOPSO as shown in Figure 3). For low number of objectives and parameters, all the methods perform equally good (first row in Figure 3). For the number of objectives more than 5, DMOPSO with the tournament selection performs the best. Among the selection mechanisms, the Fitness Proportional (FP) selection performs the worst comparing to the others. In these experiments, we observe that the quality of the solutions has been improved by the distance based ranking method in terms of the S values. Following the above experiments, we select the best results from MOPSO (DMOPSO, Tur(5)) and compare them with the results of NSGA-II and the random

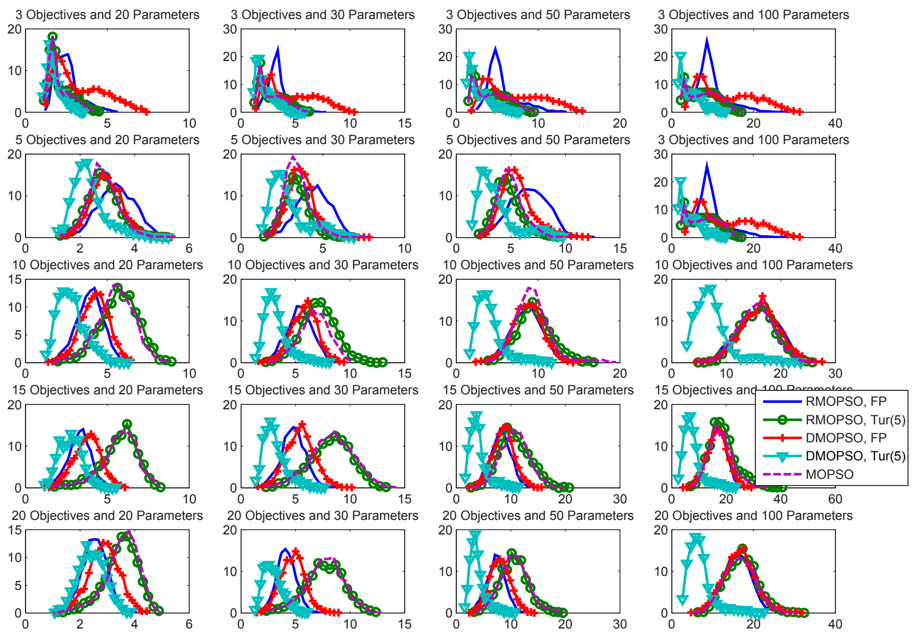


Fig. 3. Histogram of the S values for the DMOPSO and RMOPSO with FP and Tur(5) and MOPSO for different numbers of parameters and objectives

search methods (Figure 4). Here, DMOPSO obtains solutions with the same or comparable quality to the results of NSGA-II, when having low number of objectives (first row in Figure 4). For high number of objectives, DMOPSO is even performing much better than NSGA-II and the random method.

The experiments show that MOPSO methods (even the typical MOPSO) perform very well for low number of objectives and any desirable number of parameters. These results are improved for high number of objectives, as we apply a ranking mechanism to select the global particles such as in DMOPSO with tournament selection. The distance based ranking, as expected performs very well and in fact improves the results of MOPSO comparing to the AR ranking. The main issues in the above experiments are the evaluation mechanism and the fact that we have selected one test function. There are a lot of evaluation mechanisms [4,6] such as hypervolume metric, cmetric, generational distance. Note that all of these metrics are computationally intensive as we increase the number of objectives and could not be performed in a reasonable time on the huge amount of results produced in this paper. For having informative and reasonable baseline, we selected the random search method and the NSGA-II to compare the quality and the improvements in the experiments. We selected the proposed test function because its optimal front has a simple hyper-sphere shape with no specific local optima and it is very well scalable. The Pareto-front is located in the middle of the parameter space which helps us to investigate the performance of the MOPSO methods and not other aspects such as boundary handling methods.

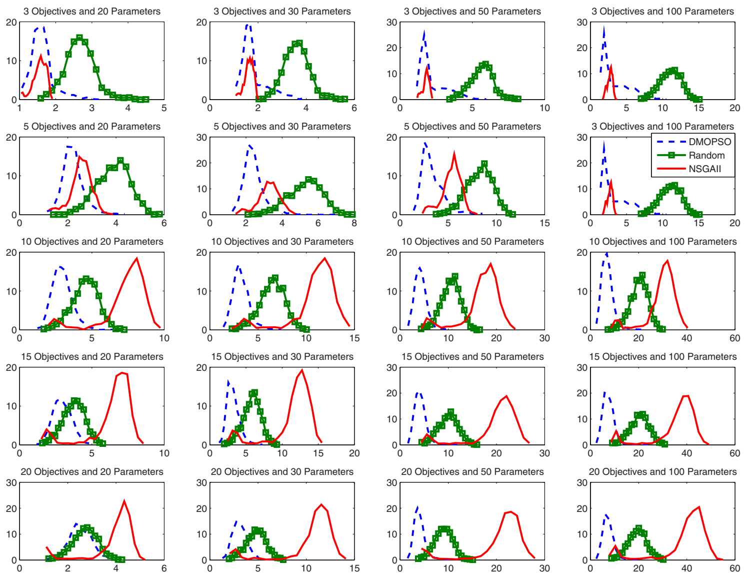


Fig. 4. Histogram of the *S* values for the DMOPSO, random and NSGA-II for different numbers of parameters and objectives

6 Conclusions and Future Work

In this paper, we study Multi-Objective Particle Swarm Optimization methods for solving problems with many ($\gg 3$) objectives and parameters. We observe that a typical MOPSO performs equally well or sometimes even worse than random search for such problems. Here, we study the performance of MOPSO by using two ranking methods in selecting the global best particles. We improve the ranking method from the literature by considering the distances between the objective values of the solutions and call it distance based ranking. The results of the MOPSO variants are compared with the results of the random search and NSGA-II methods. From our experiments, we conclude that the distance based ranking MOPSO can solve the many-objective problem better than the other methods, even for the large number of parameters. In future, we will apply the new MOPSO variants to other test problems and study other evaluation mechanisms.

References

1. Alvarez-Benitez, J., Everson, R., Fieldsend, J.: A MOPSO algorithm based exclusively on Pareto dominance concepts. In: Evolutionary Multi-Criterion Optimization, pp. 459–473 (2005)
2. Bentley, P., Wakefield, J.: Finding acceptable Pareto-optimal solutions using multiobjective genetic algorithms. In: Soft Computing in Engineering Design and Manufacturing. Springer, Heidelberg (1997)
3. Branke, J., Mostaghim, S.: About selecting the personal best in multi-objective particle swarm optimization. In: Parallel Problem Solving from Nature, pp. 523–532 (2006)

4. Coello Coello, C., Van Veldhuizen, D., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Dordrecht (2002)
5. Corne, D., Knowles, J.: Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In: Genetic and Evolutionary Computation Conference (2007)
6. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley, Chichester (2001)
7. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Parallel Problem Solving from Nature, pp. 849–858 (2000)
8. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Congress on Evolutionary Computation, pp. 825–830 (2002)
9. di Pierro, F., Djordjevic, S., Khu, S., Savic, D., Walters, G.: Automatic calibration of urban drainage model using a novel multi-objective GA. In: Urban Drainage Modelling, pp. 41–52 (2004)
10. Drechsler, D., Drechsler, R., Becker, B.: Multi-objective optimisation based on relation favour. In: Evolutionary Multi-Criterion Optimization, pp. 156–166 (2001)
11. Engelbrecht, A.: Fundamentals of Computational Swarm Intelligence. John Wiley, Chichester (2006)
12. Fieldsend, J., Singh, S.: A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In: The U.K. Workshop on Computational Intelligence, pp. 34–44 (2002)
13. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
14. Knowles, J., Corne, D.: Quantifying the effects of objective space dimension in evolutionary multiobjective optimization. In: Evolutionary Multi-Criterion Optimization, pp. 757–771 (2007)
15. Li, X.: A non-dominated sorting particle swarm optimizer for multiobjective optimization. In: Genetic and Evolutionary Computation, pp. 37–48 (2003)
16. Li, X.: Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function. In: Genetic and Evolutionary Computation, pp. 117–128 (2004)
17. Lovberg, M., Krink, T.: Extending particle swarm optimization with self-organized criticality. In: Conference on Evolutionary Computation, pp. 1588–1593 (2002)
18. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization. In: IEEE Swarm Intelligence Symposium (2003)
19. Purshouse, R.: On the Evolutionary Optimisation of Many Objectives. PhD thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, UK (2003)
20. Purshouse, R., Fleming, P.: Evolutionary multi-objective optimisation: An exploratory analysis. In: Congress on Evolutionary Computation, pp. 2066–2073 (2003)
21. Reyes-Sierra, M., Coello Coello, C.: Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)
22. Xie, X., Zhang, W., Yang, Z.: Adaptive particle swarm optimization on individual level. In: The Sixth International Conference on Signal Processing, pp. 1215–1218 (2002)

A Study of Convergence Speed in Multi-objective Metaheuristics

A.J. Nebro[1], J.J. Durillo[1], C.A. Coello Coello[2], F. Luna[1], and E. Alba[1]

[1] Department of Computer Science, University of Málaga, Spain
{antonio,durillo,flv,eat}@lcc.uma.es
[2] Department of Computer Science, CINVESTAV-IPN, Mexico
ccoello@cs.cinvestav.mx

Abstract. An open issue in multi-objective optimization is designing metaheuristics that reach the Pareto front using a low number of function evaluations. In this paper, we adopt a benchmark composed of three well-known problem families (ZDT, DTLZ, and WFG) and analyze the behavior of six state-of-the-art multi-objective metaheuristics, namely, NSGA-II, SPEA2, PAES, OMOPSO, AbYSS, and MOCell, according to their convergence speed, i.e., the number of evaluations required to obtain an accurate Pareto front. By using the hypervolume as a quality indicator, we measure the algorithms converging faster, as well as their hit rate over 100 independent runs. Our study reveals that modern multi-objective metaheuristics such as MOCell, OMOPSO, and AbYSS provide the best overall performance, while NSGA-II and MOCell achieve the best hit rates.

1 Introduction

Many real-world optimization problems require to optimize more than one objective function at the same time. These problems are called Multi-objective Optimization Problems (MOPs). Contrary to single-objective optimization problems, the solution of MOPs is not given by a single solution, but by a set of nondominated solutions called the Pareto optimal set. A solution that belongs to this set is said to be a Pareto optimum and, when the solutions of this set are plotted in the objective space, they are collectively known as the Pareto front. Obtaining the Pareto front is the main goal in multi-objective optimization. Additionally, many real-world MOPs typically need computationally expensive methods for computing the objective functions and constraints. In this context, deterministic techniques are generally not applicable, which leads therefore to using approximate methods [7]. Among them, metaheuristics [1,7] are nowadays used extensively to deal with MOPs.

The performance of these algorithms is normally assessed using benchmarks, such as the Zitzler-Deb-Thiele (ZDT) test suite [19], the Deb-Thiele-Laumanns-Zitzler (DTLZ) problem family [3], and the Walking-Fish-Group (WFG) test problems [9]. The experimentation methodology typically lies in computing a pre-fixed number of function evaluations and then comparing the obtained results by considering different quality indicators [11].

The motivation driving us is that the objective functions of many real-world problems are hard to compute, so applying metaheuristics requiring a high number of function evaluations to solve them is not a satisfactory approach in practice. In this context, there are applications in which it can be more important to obtain a reasonably good approximation to the Pareto front of a given MOP *faster* than to search for a higher quality solution set but requiring *more time*. Thus, an open research area is to design techniques with this goal in mind, and some works that address this issue have recently appeared. Santana-Quintero et al. propose in [17] a PSO algorithm using rough sets theory, and it is used to solve problems using 4,000 fitness function evaluations, which is a low number compared to today's standards in the specialized literature. In a related paper, Hernández-Díaz et al. [8] propose a hybrid algorithm between a multi-objective differential evolution approach and rough sets theory, which only performs 3,000 fitness function evaluations. In [18], a more efficient multi-objective PSO algorithm is presented; this algorithm is able to provide accurate Pareto fronts of MOPs computing only 2,000 fitness function evaluations. Eskandari et al. explore in [6] the use of dynamic population sizing to design an algorithm called FastPGA, which outperforms NSGA-II when computing less than 10,000 solution evaluations. Knowles studies multi-objective optimization calculating only 260 function evaluations [10]; he proposes an algorithm called ParEGO, which outperforms NSGA-II using such a low number of evaluations.

In this paper, we are interested in analyzing the convergence speed of six state-of-the-art multi-objective metaheuristics to give hints about their efficiency when solving 21 MOPs comprising the test suites ZDT, DTLZ, and WFG. The algorithms are two Genetic Algorithms (NSGA-II [2], and SPEA2 [20]), an Evolution Strategy (PAES [12]), a Particle Swarm Optimization algorithm (OMOPSO [16]), a Scatter Search method (AbYSS [15]), and a cellular Genetic Algorithm (MOCell [13]). In our study, to assess the quality of a front we have employed the hypervolume quality indicator [21]. To assure that an algorithm has successfully solved a problem it needs reaching a fixed percent of the hypervolume of the true Pareto front. In Fig. 1 we show different fronts obtained for problem ZDT1 with different percentages of hypervolume. We can observe that the front with a hypervolume of 98.26% represents a reasonable approximation to the true Pareto front in terms of convergence and diversity of solutions. So, we have taken 98% of the hypervolume of the true Pareto front as a criterion to consider that a MOP has been successfully solved. Thus, those algorithms requiring fewer function evaluations to achieve this termination condition can be consider to be *faster*. Using the hypervolume in the stopping condition also allows us to obtain a hit rate for the algorithms, i.e., their percentage of successful executions.

The rest of the paper is organized as follows. Section 2 is devoted to the experimentation, including the parameter settings and the methodology adopted in the tests. Section 3 provides an analysis of the obtained results. The conclusions and potential lines for future work are presented in Section 4.

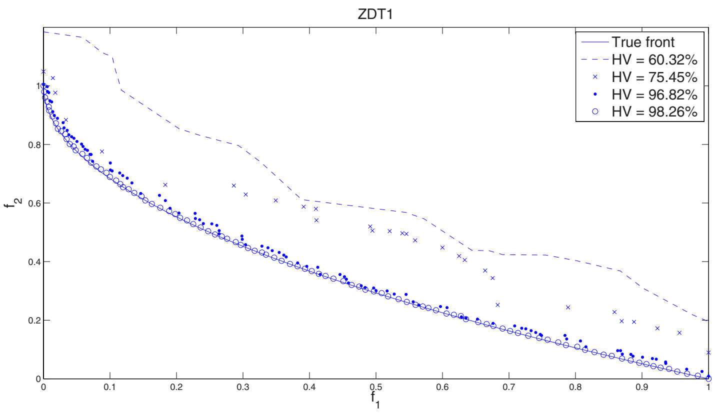


Fig. 1. Fronts with different hypervolume values obtained for problem ZDT1

2 Experimentation

As commented before, to carry out our study we have selected the two most widely known and used metaheuristics in the field: NSGA-II [2] and SPEA2 [20], and we have compared them to other classical algorithm: PAES [12], and to three other modern algorithms: OMOPSO [16], AbYSS [15], and MOCell [13] (in particular, we have used the asynchronous variant named aMOCell4 in [14]). We do not describe them here due to space restrictions. Authors unfamiliar with these techniques should revise the indicated references. We have used the implementation of these algorithms provided by jMetal [5], a Java-based framework for developing metaheuristics for solving multi-objective optimization problems.

The benchmarking MOPs used to evaluate the six metaheuristic algorithms have been the aforementioned ZDT [19], DTLZ [3], and WFG [9] test suites. The two latter families of MOPs have been used with their bi-objective formulation.

2.1 Parameterization

We have chosen a set of parameter settings that aims at guaranteeing a fair comparison among the algorithms. All the evolutionary algorithms (NSGA-II, SPEA2, and MOCell) use an internal population of size equal to 100; OMOPSO is configured with 100 particles and with a maximum number of 100 leaders also. The size of the archive in PAES is 100 as well. AbYSS uses a population size of 20, which is also the size of the RefSet; the size of the external archive is 100. In MOCell a toroidal grid of 100 individuals (10×10) has been chosen for structuring the population, and an archive of 100 individual is used.

For the genetic algorithms, we have used SBX and polynomial mutation [4] as operators for crossover and mutation, respectively. The distribution indexes for both operators are $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability is $p_c = 0.9$ and the mutation probability is $p_m = 1/n$, where n is the number of decision variables. In PAES, we have also used the polynomial mutation operator, with the same distribution index. OMOPSO makes use of two types of mutation operators: uniform and non-uniform. AbYSS uses polynomial mutation in the local search procedure and SBX in its solution recombination method.

2.2 Methodology

Since our main interest is to analyze the convergence speed of the studied metaheuristics, we have to measure the number of evaluations needed by the algorithms to achieve a hypervolume equal or higher than the 98% of the value of that quality indicator when applied to the Pareto front of the considered problem.

In our experiments, each algorithm is executed until a maximum of 1,000,000 function evaluations have been performed. At every 100 evaluations (that is, at each iteration in the population-based metaheuristics), we measure the hypervolume of the nondominated solutions found so far. Therefore, in NSGA-II and SPEA2 we have considered the nondominated solutions at each generation, whereas in PAES, AbYSS, and MOCell, we have used the external population and, in OMOPSO, the leaders archive. We consider as stopping condition either reaching the desired hypervolume value or computing the 1,000,000 evaluations previously indicated.

Using the hypervolume as the stopping condition allows us to obtain a *hit rate* for the algorithms, i.e., the percentage of successful executions. An execution is successful if the metaheuristic stops before reaching 1,000,000 function evaluations.

We have performed 100 independent runs of each algorithm for each problem instance. Since we are dealing with stochastic algorithms, we need to perform a statistical analysis of the obtained results in order to compare them with a certain level of confidence. Next, we describe the statistical tests that we have carried out for ensuring such statistical confidence. First, a Kolmogorov-Smirnov test is performed in order to check whether the values of the results follow a normal (Gaussian) distribution or not. If they follow a normal distribution, the Levene test checks for the homogeneity of the variances. If the samples have equal variance (positive Levene test), an ANOVA test is done; otherwise, we perform a Welch test. For non-Gaussian distributions, the non-parametric Kruskal-Wallis test is used in order to compare the medians of the algorithms. We always consider in this work a confidence level of 95% (i.e., a significance level of 5% or p-value under 0.05) in the statistical tests, which means that the differences are unlikely to have occurred by chance with a probability of 95%. Successful tests are marked with the '+' symbol in the last column in the table containing the results (see Table 1); conversely, a '-' symbol means that no statistical confidence was found (p-value > 0.05). Looking for homogeneity in the presentation of the results, we use the median, $\tilde{x}$, and interquartile range, IQR, as measures of location (or central tendency) and statistical dispersion, respectively, since some samples are normal and others are not. We have also performed a post-hoc testing phase (not included in the paper because of space constraints) using the `multcompare` function provided by Matlab©, which allows for a multiple comparison of samples. In general, it can be said that the differences of the best algorithms with respect of the others for each MOP are statistically significant at 95% of confidence level.

Table 1. Median and *IQR* of the number of evaluations computed by the algorithms

	NSGA-II	SPEA2	PAES	OMOPSO	AbYSS	MOCell	
Problem	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	$\bar{x}_{IQR}$	
ZDT1	1.43e+4 $_{8.0e+2}$	2.12e+4 $_{1.6e+3}$	1.32e+4 $_{1.1e+4}$	6.80e+3 $_{2.0e+3}$	1.37e+4 $_{1.6e+3}$	1.30e+4 $_{1.2e+3}$	+
ZDT2	2.43e+4 $_{1.8e+3}$	–	1.71e+5 $_{2.0e+5}$	8.90e+3 $_{3.6e+3}$	1.71e+4 $_{2.8e+3}$	1.17e+4 $_{4.0e+3}$	+
ZDT3	1.27e+4 $_{9.0e+2}$	1.72e+4 $_{1.5e+3}$	2.56e+4 $_{2.3e+4}$	9.85e+3 $_{2.7e+3}$	1.27e+4 $_{2.0e+3}$	1.30e+4 $_{1.3e+3}$	+
ZDT4	2.13e+4 $_{5.0e+3}$	2.46e+5 $_{2.6e+5}$	4.41e+4 $_{1.8e+4}$	–	2.28e+4 $_{1.1e+4}$	1.63e+4 $_{5.0e+3}$	+
ZDT6	2.88e+4 $_{1.2e+3}$	5.27e+4 $_{5.5e+3}$	9.95e+3 $_{1.2e+4}$	2.80e+3 $_{1.5e+3}$	1.56e+4 $_{1.2e+3}$	2.09e+4 $_{1.3e+3}$	+
DTLZ1	2.51e+4 $_{9.4e+3}$	–	8.73e+4 $_{1.3e+5}$	1.00e+6 $_{4.7e+4}$	2.37e+4 $_{1.2e+4}$	2.01e+4 $_{7.7e+3}$	+
DTLZ2	8.10e+3 $_{1.2e+3}$	–	3.07e+4 $_{2.0e+4}$	8.20e+3 $_{3.1e+3}$	4.70e+3 $_{9.0e+2}$	5.60e+3 $_{9.0e+2}$	+
DTLZ3	1.18e+5 $_{5.7e+4}$	–	1.00e+6 $_{2.7e+5}$	–	1.19e+5 $_{7.5e+4}$	6.73e+4 $_{2.3e+4}$	+
DTLZ4	8.50e+3 $_{1.4e+3}$	–	–	1.25e+4 $_{3.8e+3}$	4.80e+3 $_{7.5e+2}$	1.00e+6 $_{9.9e+5}$	+
DTLZ5	7.95e+3 $_{1.1e+3}$	–	3.14e+4 $_{2.9e+4}$	8.45e+3 $_{2.9e+3}$	4.65e+3 $_{8.0e+2}$	5.80e+3 $_{8.5e+2}$	+
DTLZ6	1.00e+6 $_{9.7e+5}$	–	7.89e+4 $_{1.5e+5}$	4.10e+3 $_{1.5e+3}$	–	–	+
DTLZ7	1.36e+4 $_{1.0e+3}$	2.35e+4 $_{2.6e+3}$	–	6.15e+3 $_{2.6e+3}$	1.06e+4 $_{1.7e+3}$	1.11e+4 $_{1.6e+5}$	+
WFG1	4.38e+4 $_{1.1e+5}$	2.27e+5 $_{8.2e+5}$	–	–	–	4.16e+4 $_{1.7e+4}$	+
WFG2	1.75e+3 $_{4.5e+2}$	2.40e+3 $_{8.0e+2}$	1.32e+5 $_{1.6e+5}$	1.80e+3 $_{4.0e+2}$	1.85e+3 $_{2.4e+3}$	1.40e+3 $_{8.0e+2}$	+
WFG3	–	–	–	–	–	–	-
WFG4	1.84e+4 $_{6.2e+3}$	–	–	2.23e+5 $_{1.3e+5}$	6.75e+3 $_{2.4e+3}$	1.05e+4 $_{3.1e+3}$	+
WFG5	–	–	–	–	–	–	-
WFG6	1.00e+6 $_{5.2e+5}$	9.05e+4 $_{8.0e+4}$	–	7.30e+3 $_{1.2e+3}$	–	1.00e+6 $_{5.5e+5}$	+
WFG7	1.62e+5 $_{2.7e+5}$	–	–	1.49e+4 $_{2.6e+3}$	9.60e+3 $_{3.4e+3}$	1.21e+4 $_{3.4e+3}$	+
WFG8	–	–	–	–	–	–	+
WFG9	–	–	–	8.93e+4 $_{4.9e+4}$	–	–	+

3 Analysis of Results

In this section, we analyze the obtained results. Table 1 shows the median and the interquartile range of the number of evaluations needed by the different optimizers when solving all the problems. When an optimizer is not able to reach acceptable fronts upon performing 1,000,000 function evaluations after the 100 independent runs, its result appears as '–', and it is not taken into account in the statistical tests. Concretely, the '–' symbol means that the median of the number of function evaluations is 1,000,000 and the IQR is 0. However, it is worth mentioning that the *IQR* only considers the values between the 25th and the 75th percentiles, so it is possible that the algorithm was successful only in a few executions (less than 25% of the independent runs executed). To ease the analysis of the results in Table 1, the cells containing the lowest number of function evaluations have a grey colored background. There are two grey levels: the darker grey indicates the best (lowest) value, while the lighter grey points out the second best value.

The hit rates are reported in Table 2. A '$\surd$' in a cell means a 100% hit rate, while a '–' indicates that the problem could not be solved in none of the 100 independent runs.

3.1 ZDT Problems

We start by analyzing the results obtained when solving the ZDT test suite, which is composed of five MOPs having different properties (convex, non-convex, disconnected, multi-frontal, non uniformly spaced). In this benchmark OMOPSO is the fastest optimizer, requiring the lowest number of evaluations to reach the stopping condition in four out of the five MOPs of the ZDT family. Furthermore,

Table 2. Hit Rate

Problem	NSGA-II	SPEA2	PAES	OMOPSO	AbYSS	MOCell
ZDT1	√	√	√	√	√	√
ZDT2	√	–	0.99	√	√	√
ZDT3	√	√	√	√	√	√
ZDT4	√	0.99	√	–	√	√
ZDT6	√	√	0.96	√	√	√
DTLZ1	√	–	0.91	0.28	√	√
DTLZ2	√	–	√	√	√	√
DTLZ3	√	–	0.30	0.01	√	√
DTLZ4	0.89	–	–	√	√	0.38
DTLZ5	√	–	√	√	√	√
DTLZ6	0.40	–	0.97	√	0.19	0.10
DTLZ7	0.99	√	0.16	√	0.89	0.76
WFG1	0.83	0.73	–	–	0.21	√
WFG2	√	√	0.99	√	0.98	√
WFG3	–	–	–	–	0.17	–
WFG4	√	–	–	√	√	√
WFG5	–	–	–	–	0.11	–
WFG6	0.34	–	√	√	0.13	0.37
WFG7	0.99	–	–	√	√	√
WFG8	–	–	–	–	0.18	0.12
WFG9	–	–	–	0.99	0.24	0.24

except for the ZDT4 problem, the differences are noticeable when compared with the second best performer, particularly in ZDT1 and ZDT2. Despite its good global results, OMOPSO is the only metaheuristic unable to solve ZDT4, a multi-frontal problem, in less than 1,000,000 function evaluations. MOCell, the cellular GA, is the fastest metaheuristic solving ZDT4, and both NSGA-II and MOCell are the second fastest algorithms in two problems. PAES, the simplest one of the compared algorithms, obtains the second best value in ZDT6. Concerning SPEA2 and AbYSS, they do not obtain the best nor the second best results in any problem.

When examining the hit rate results (see Table 2), we see that NSGA-II, AbYSS, and MOCell achieve a 100% for all the problems, while SPEA2 and OMOPSO fail in problems ZDT2 and ZDT4, respectively. We observe that PAES has a hit rate of 0.96 on problem ZDT6, so it has been unable to find the Pareto front of the problem in four out of the 100 independent runs.

3.2 DTLZ Test Problems

The DTLZ test suite is composed of seven MOPs, some of them including properties not found in any of the problems from the ZDT family. For example, DTLZ1 is a linear problem, and both DTLZ5 and DTLZ6 have degenerate Pareto fronts (when the number of objectives is greater than two).

If we focus on convergence speed, AbYSS is the fastest algorithm at reaching 98% the hypervolume of the true Pareto fronts on three out of the seven MOPs from this benchmark, and the second fastest in other two. The second algorithm is MOCell, which requires the lowest number of evaluations on DTLZ1 and DTLZ3 and it also is the second fastest solver in two out of the seven problems. OMOPSO obtains the best results in two problems: DTLZ6 and DTLZ7.

We examine now the hit rate of the metaheuristics. According to Table 2, it is noticeable that SPEA2 is only able to solve the DTLZ7 problem in less than

1,000,000 functions evaluations. It is also remarkable that OMOPSO only finds an accurate front in one of the 100 independent runs performed on problem DTLZ3. Conversely, according to the hit rates obtained, NSGA-II and AbYSS appear as the most successful algorithms on the DTLZ problem family, followed by MOCell.

3.3 WFG Test Problems

The WFG test suite is composed of nine MOPs having different properties. A first look at the number of evaluations required by the six studied metaheuristics shows that none of them is able to provide accurate Pareto fronts of three problems (WFG3, WFG5, and WFG8) in less than 1,000,000 function evaluations, and many algorithms have difficulties when solving the others MOPs. This clearly indicates that this benchmark is harder to be solved than both the ZDT and the DTLZ problem families.

Proceeding as in the two previous benchmarks, we start by analyzing the convergence speed. The fastest algorithm is MOCell, which requires the lowest number of evaluations in two cases, having the second best behavior on two of the WFG problems. OMOPSO and AbYSS achieve the best results in two out of the nine MOPs. NSGA-II obtains the second lowest number of evaluations in two problems and SPEA2 does the same in one.

The technique providing the highest hit rates is MOCell, which achieves a 100% on WFG1, WFG2, WFG4, and WFG7, and hit rates of 0.37, 0.12, and 0.24 on WFG6, WFG8 and WFG9, respectively. MOCell is followed by AbYSS (100% on WFG4 and WFG7), which is the only solver able of finding some accurate fronts in problems WFG3 and WFG5. It is remarkable the behavior of OMOPSO, which obtains a 100% hit rate in practically five problems, but it fails in the other four MOPs. The worst ranked algorithms according to the hit rate for this benchmark are SPEA2 and PAES. As commented before, those results below a hit rate of 25% are reported in Table 1 as '–', as it happens with AbYSS and WFG1, WFG3, WFG5, WFG6, WFG8, and WFG9.

3.4 Discussion

If we merely make a global ranking of the fastest algorithms in our study (see Table 3), it would be led by MOCell (five best results, six second best ones), OMOPSO (eight best results), and AbYSS (five best results, two second best ones). The hit rate ranking would be headed by NSGA-II. It performs the best in the DTLZ test suite, as MOCell and AbYSS do in the WFG family. In the case of the ZDT problems, these three approaches perform equally well. It is worth mentioning that AbYSS is the only metaheuristic providing a hit rate higher than 0% in all the problems.

These conclusions are relevant, and we believe that they are the most important contributions of our study. However, although from a practical point of view the hints of the type "if you want to solve a problem fast, try first MOCell and OMOPSO" or "if you want the highest chance to solve a MOP, try NSGA-II", can be useful, it would be more interesting to provide some hints, given the characteristics of a given MOP, regarding the algorithm that is more suitable to solve

Table 3. Ranking of the algorithms

Rank	Convergence speed	Hit rate
1	MOCell	NSGA-II
2	OMOPSO	MOCell
3	AbYSS	AbYSS
4	NSGA-II	OMOPSO
5	PAES	PAES
6	SPEA2	SPEA2

it. The three benchmarks we have used provide us with a range set of problems, having each of them different features. Unfortunately, their analysis in [9] indicate that it is far from simple to make a clear classification of the 21 problems according to their properties (convex, concave, linear, disconnected, multifrontal, separable, etc). In any case, we attempt to draw some (more general) conclusions based on our study, subject to the evident limitations previously indicated.

If we focus on the multi-modality feature, it is present in the following problems: ZDT3, ZDT4, ZDT6, DTLZ1, DLTZ3, DTLZ7, WFG4, and WFG9 (this is also deceptive) [9]. An analysis of the evaluations required to solve these MOPs shows that MOCell is the fastest algorithm in ZDT4, DTLZ1, and DTLZ3, and the second fastest in WFG4. OMOPSO is the fastest algorithm to solve problems ZDT3, ZDT6 and DTLZ7, and it is the only solver able to achieve a hit rate of 99% on problem WFG9, but it fails when solving ZDT4, DTLZ1, and DTLZ3. According to these results, it is not clear whether OMOPSO should be discarded or not when dealing with this type of problems.

There are three problems having disconnected Pareto fronts: ZDT3, DTLZ7, and WFG2. Our study reveals that OMOPSO is the fastest in the first two and the third fastest in the last one. The second algorithm is MOCell, which requires the lowest number of evaluations in problem WFG2, followed by NSGA-II which is the second fastest in problems ZDT3 and WFG2.

4 Conclusions and Future Work

We have evaluated six metaheuristics over a set of 21 MOPs in order to study the performance of the algorithms concerning their convergence speed, i.e., their velocity to reach an accurate Pareto front using a stopping condition based on the hypervolume of the true Pareto front. We have also evaluated the percentage of successful executions or hit rate.

In the context of the problems analyzed, the experimentation methodology, and the parameter settings used, we can state that, regarding convergence speed, MOCell, OMOPSO, and AbYSS are the most competitive algorithms, followed by NSGA-II and PAES. SPEA2 is the last algorithm in the ranking, so it can be considered as the slowest of the compared techniques.

As to the hit rate, NSGA-II is the most salient algorithm in the DTLZ test suite, followed by AbYSS and MOCell. In the WFG family, AbYSS and MOCell are the first one in the ranking, followed by NSGA-II. The algorithms providing the worst behavior are SPEA2 (fails in 14 problems) and PAES (fails in 8 problems).

Taking into account problem properties, we have found out that OMOPSO performs the best in disconnected problems. Concerning multi-modality, MOCell provides good values, while OMOPSO has an "all or nothing" behavior: it is either among the best when solving a problem, or it has a hit rate of zero (i.e., it is the worst).

We have presented a first study of the behavior of multi-objective metaheuristics concerning their convergence speed as well as their hit rates. A line of future work is to deepen in the study of the features of the problems, to try to determine more precisely which algorithms are more suited to solve a certain type of MOP. Other interesting research line is to analyze the best parameter settings of the algorithms in order to make them to converge faster while maintaining a high degree of success when solving the problems.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Education and Science and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project). Juan J. Durillo is supported by grant AP-2006-03349 from the Spanish Ministry of Education and Science. Carlos A. Coello Coello acknowledges support from CONACYT project no. 45683-Y.

References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35(3), 268–308 (2003)
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
3. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, Heidelberg (2005)
4. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
5. Durillo, J.J., Nebro, A.J., Luna, F., Dorronsoro, B., Alba, E.: jMetal: a java framework for developing multi-objective optimization metaheuristics. Technical Report ITI-2006-10, Dpto. de Lenguajes y Ciencias de la Computación, University of Málaga (2006)
6. Eskandari, H., Geiger, C.D., Lamont, G.B.: FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 141–155. Springer, Heidelberg (2007)
7. Glover, F., Kochenberger, G.A.: Handbook of Metaheuristics. Kluwer Academic Publishers, Dordrecht (2003)
8. Hernández-Díaz, A.G., Santana-Quintero, L.V., Coello Coello, C., Caballero, R., Molina, J.: A New Proposal for Multi-Objective Optimization using Differential Evolution and Rough Sets Theory. In: Keijzer, M., et al. (eds.) Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 675–682 (2006)

9. Huband, S., Hingston, P., Barone, L., While, L.: A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE Transactions on Evolutionary Computation 10(5), 477–506 (2006)
10. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1), 50–66 (2006)
11. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
12. Knowles, J.D., Corne, D.W.: The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In: Congress on Evolutionary Computation, pp. 98–105 (1999)
13. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: A cellular genetic algorithm for multiobjective optimization. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2006), pp. 25–36 (2006)
14. Nebro, A.J., Durillo, J.J., Luna, F., Dorronsoro, B., Alba, E.: Design issues in a multiobjective cellular genetic algorithm. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 126–140. Springer, Heidelberg (2007)
15. Nebro, A.J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J.J., Beham, A.: AbYSS: Adapting scatter search to multiobjective optimization. IEEE Transactions on Evolutionary Computation (to appear, 2008)
16. Reyes Sierra, M., Coello Coello, C.A.: Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)
17. Santana-Quintero, L.V., Ramírez-Santiago, N., Coello Coello, C.A., Molina Luque, J., García Hernández-Díaz, A.: A New Proposal for Multiobjective Optimization Using Particle Swarm Optimization and Rough Sets Theory. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN IX 2006. LNCS, vol. 4193, pp. 483–492. Springer, Heidelberg (2006)
18. Toscano-Pulido, G., Coello Coello, C.A., Santana-Quintero, L.V.: EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 272–285. Springer, Heidelberg (2007)
19. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
20. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: EUROGEN 2001, pp. 95–100 (2002)
21. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)

Team Algorithms Based on Ant Colony Optimization – A New Multi-Objective Optimization Approach

Christian Lezcano, Diego Pinto, and Benjamín Barán

Polytechnical School - National University of Asunción
P.O. Box 2111 - Paraguay
{clezcano,dpinto,bbaran}@pol.una.py
http://www.una.py

Abstract. This paper proposes a novel Team Algorithm (TA) approach based on Ant Colony Optimization (ACO) for multi-objective optimization problems. The proposed method has shown a significant cooperative effect of different algorithms combined in a team of algorithms, achieving robustness in the resolution of a set of various combinatorial problems. Experimentally, the proposed approach has verified a balance on different performance measures in problems as the Traveling Salesman Problem (TSP), the Quadratic Assignment Problem (QAP) and the Vehicle Routing Problem with Time Windows (VRPTW). Robustness and balance are achieved due to a novel classification and selection of the algorithms to be used by the team, considering Pareto concept.

Keywords: Team Algorithms (TA), Ant Colony Optimization (ACO) and Multi-objective Optimization Problem (MOP).

1 Introduction

A Multi-objective Optimization Problem (MOP) can be defined as the problem of finding a set of solutions that satisfies given constraints and optimizes several objective functions simultaneously. Usually, these objective functions are in conflict with each other [1]. MOP is widely treated with different optimization paradigms [1-4], as Multi-Objective Evolutionary Algorithm – MOEA, which have been successfully applied to solve highly complex problems. On the other hand, the "*No Free Lunch* - NFL" theorem [5] states that on average, all algorithms have the same performance. Considering the NFL theorem, the development of Team Algorithms (TA) [6] is a valid alternative for achieving high *robustness* on average.

This paper proposes a novel TA approach based on Multi-Objective Ant Colony Optimization (MOACO) algorithms for solving combinatorial optimization problems in a multi-objective context. Basically, the new proposal tries to improve the state of the art in this area [6]. Experimental tests were carried out on different bi-objective instances of the Traveling Salesman Problem (TSP), the Quadratic Assignment Problem (QAP) and the Vehicle Routing Problem with Time Windows (VRPTW). To measure an algorithm's quality in a MOP context, various multi-objective performance figures (measures) were considered, such as distance, distribution, extension, error and quality of solutions [7, 8].

2 Multi-Objective Optimization

A Multi-Objective Optimization Problems (MOP) generally consists of a set of *n* decision variables, a set of *k* objective functions and a set of *m* restrictions [1]. Objective functions and restrictions are function of the decision variables. Therefore, MOP generally optimizes:

$$\mathbf{z} = f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) \tag{1}$$

subject to
$$g(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) \leq 0 \tag{2}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \boldsymbol{X}$ is a decision vector, $\boldsymbol{X}$ denotes the decision space of $\boldsymbol{f(x)}$, $\boldsymbol{z} = (z_1, z_2, \ldots, z_k) \in \boldsymbol{Z}$ is an objective vector while $\boldsymbol{Z}$ denotes the objective space of $\boldsymbol{f(x)}$. The *feasible solution set* $\Omega \subset X$ is defined as a set of decision vectors $\boldsymbol{x}$ that satisfies $g(\boldsymbol{x})$. A vector $\boldsymbol{u} \in \Omega$ is said to *dominate* $\boldsymbol{v} \in \Omega$ (denoted by $\boldsymbol{u} \succ \boldsymbol{v}$) if $\boldsymbol{u}$ is at least as good as $\boldsymbol{v}$ in every objective function and strictly better in at least one objective function. For a given MOP, the *Pareto optimal set* P* is defined as the set of non-dominated solutions of Ω. The objective space of P*, known as *Pareto Front*, is denoted as F*, i.e. F*=$\boldsymbol{f}$(P*).

Performance figures are used in this work with two aims: *(i)* to compare different TA approaches in a multi-objective context, and *(ii)* to measure performance of slave processes for the selection of algorithms to be used in the next iteration of a TA. The performance figures used in this work are presented in Table 1.

Table 1. Performance measures

Name	*Symbol*	*Ref.*
Generational Distance	M_1	[8]
Distribution	M_2	[8]
Extension	M_3	[8]
Error	M_4	[9]
ONVGR†	M_5	[9]

†Overall Non-dominated Vector Generation Ratio

3 Multi-Objective Ant Colony Optimization

Multi-Objective Ant Colony Optimization (MOACO) is a search technique inspired by the natural behavior of ant colonies [2, 3]. Different studies demonstrated empirically the efficiency of MOACOs in solving a MOP [2, 3, 6, and 14-20]. In this paper we considered the use of 9 MOACO algorithms (see Table 2) representing the state of the art in this area.

Table 2. MOACO algorithms

Symbol	*Algorithms*	*Symbol*	*Algorithms*
MAS	*Multi-objective Ant System* [6]	COMP	*COMPETants* [16]
BIANT	*Bi-criterion Ant* [14]	MOACS	*Multi-Objective Ant Colony System* [17]
BIMC	*Bi-criterion Multi Colony* [14]	M3AS	*Multi-objective Max-Min Ant System* [18]
PACO	*Pareto-Ant Colony Optimization* [15]	MOAQ	*Multiple Objective Ant Q* [19]
MOA	*Multi-objective Omicron ACO* [20]		

4 Team Algorithms

Team Algorithm (TA) is a technique used to take advantage of parallel computational resources in an asynchronous environment, such as a computer network, combining different algorithms to solve the same problem [6, 9, 10]. For example, each algorithm of a team can be executed in a different processor of the network. A process of a TA, usually known as Master Process, can dynamically define the mapping of algorithms to computers. Figure 1 presents this centralized master-slave model [6].

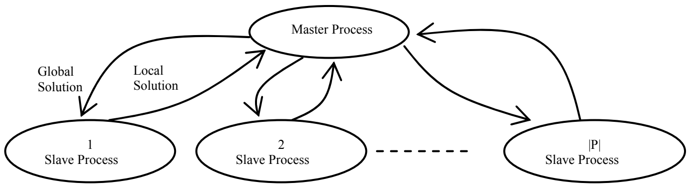


Fig. 1. Interaction between the master and each of |*P*| slaves. The slave processes send their best local solutions to the master process. The master guides the search process by sending global solutions to the slaves. The master also defines what algorithm will be executed in each slave, what may be dynamically modified.

4.1 Related Works

Evolutionary Computation was associated with the concept of parallelism since its inception [12, 13]. However, distributed models have only recently been incorporated for resolving MOPs. Fernandez et al. proposed a Team Algorithm of MOEAs (TA-MOEA) considering different MOEA algorithms running in each processor of a distributed system [10]. Moreover, several parallel approaches of ACO were proposed in [13]. Recently, Paciello et al. proposed a Team Algorithm based on MOACO algorithms (TA-MOACO) [6], where the implementation of the TA-MOACO is similar to TA-MOEA [10].

4.2 General Method

In general, each slave process looks for the best Pareto solutions of some particular region of the search space. These solutions are transmitted to the master process, which generates a set of Pareto solutions of the space explored by the TA.

Depending on the approach [6, 10], each algorithm is selected according to some criterion in order to be assigned to a slave process in the next iteration. Normally, the selection criteria are taken proportional to the quality of the contributed solutions. Typically, the quality of the contributed solutions is defined with one of the performance figures presented in Table 1. This interaction between processes (master and slaves) is carried out until a stop condition is satisfied. To facilitate the understanding of the implemented algorithms, the following nomenclature is presented:

P:	Set of available computers on the network, i.e. $P = \{p_i \mid i=1, 2,\ldots, \lvert P\rvert\}$.
H:	Set of algorithms, where $H = \{h_j \mid j=1, 2,\ldots, \lvert H\rvert\}$.
W:	Set of slave processes mapping, where $W = \{w_k\in P\times H \mid w_k = (p_i, h_j), p_i\in P, h_j\in H, k=1,2,\ldots,\lvert W\rvert\}$, note that $P\times H$ is a Cartesian product and $W \subset P\times H$.
P_k:	Pareto Set calculated by a slave process $w_k\in W$.
Γ:	Set formed by the Pareto sets P_k, i.e. $\Gamma = \{P_k \mid w_k\in W\}$.
Υ:	Pareto fronts associated to Γ, where $\Upsilon = \{F_k \mid F_k = f(P_k)\}$. Note that $\Upsilon = f(\Gamma)$.
N_k:	Pareto performance figures associated with a slave process $w_k\in W$, i.e. $N_k=\{M_r \mid r=1, 2,\ldots, \lvert N_k\rvert\}$. Note that in this work $\lvert N_k\rvert \leq 5$ (see Table 1).
Ψ:	Performance associated to W, where $\Psi = \{N_k \mid k=1, 2,\ldots, \lvert W\rvert\}$.
P_{known}:	Known Pareto set that approximates P^*.
F_{known}:	Known Pareto front that approximates F^*, where $F_{known} = f(P_{known})$.

For this work, each process w is executed on a different processor p_i, i.e., at all times $\lvert W\rvert = \lvert P\rvert$ (a slave processor only runs one algorithm at a time).

Considering the previous nomenclature, Algorithm 1 presents a generic procedure for a master process while Algorithm 2 corresponds to the procedure of the k-th slave process.

```
Algorithm 1: Generic Master Process
Input: $P, H$
Output: $P_{known}$ and $F_{known}$
Initialize configuration parameters and $P_{known} = \emptyset$
$W$ = Initial Mapping ($P, H$)
Initial Launch Slave Processes ($W$)
while (stop condition is no achieved) do
   $\Gamma$ = Receive solutions from Slaves Processes ($P$)
   Update $P_{known}$ and $F_{known}$
   $\Psi$ = Calculated Performance of Slaves ($\Upsilon$, $F_{known}$)
   $W$ = Mapping ($P, H, \Psi$)
   Launch Slave Processes ($W$, $P_{known}$, Control-Parameters)
end while
Send stop-orders to slaves
return $P_{known}$ and $F_{known}$
```

```
Algorithm 2: Generic Slave Process ($w_k$)
Input: $h_k$, $P_{known}$, Seed
Receive initial configuration from Master Process ($h_k$)
while (stop-order is not received) do
   P' = Select Solutions ($P_{known}$, Seed)
   $P_k$ = Run Algorithm ($h_k$, P')
   Send best solutions to Master Process ($P_k$)
   Receive information from Master Process ($h_k$, $P_{known}$, Control-Parameters)
end while
```

4.3 Simple-TA of MOACOs

The method proposed in [6, 10] considered a classification of slaves processes using a single figure of merit, i.e. $|M_k|=1$. In this work we call that TA-MOACO proposal *simple*-TA (*s-TA*). Basically, the classification of slave processes in s-TA consists of an ordering of slave processes from best to worst according to their performances obtained in the last iteration. Then, the algorithm used by the best slave process replaces the algorithm of the worst slave process. Thus, as the evolutionary cycle progresses, a few algorithms tend to be associated with all the processes of the team. Note that this approach is a case of *sudden death* where an algorithm with poor performance at the beginning of the execution never again is selected.

5 Proposed Methods

TAs recently proposed in the literature [6, 10] do not simultaneously considers different factors affecting convergence, distribution and extension of the quality of the solutions associated with a Pareto Optimal set. The use of a single figure of merit is a partial assessment of the actual performance of slave processes. Therefore, this work considers several figures of merit simultaneously, with the hope that it would enhance the search capabilities of a TA balancing different desirable characteristic, i.e. good values of several performance measures. Under these conditions, the master process may discern the algorithms used to really improve performance figures of those that need to be improved. Moreover, the master process could decide what portions of P_{known} would be sent to each slave process in order to promote the exploration and exploitation of various portions of the search space (see *Select Solutions* in Algorithm 2).

Considering the above-mentioned idea, this paper proposes for the first time the simultaneous use of several performance figures (in a multi-objective context) associated with the algorithms of a TA. Thus, a classification of slave processes (R) in the Pareto sense may be calculated. This classification indicates which slave processes obtained better performance and therefore should have a greater chance of being used at a next iteration. In short, the R classification is based on successive Pareto sets of slaves processes $w \in W$ considering performance figures, i.e. $R = \{R_l \mid l=1, 2,\ldots, |R|\}$ where $R_1 \succ R_2 \succ \ldots \succ R_{|R|}$. Classification by Pareto fronts is performed using an algorithm known as *fast non-dominated sorting* proposed in [21].

An important issue to be highlighted is that the classification of slave processes of *s*-TA is a particular case of the current proposal using a single performance metric. Basically, two new approaches are proposed below: *elitist*-TA (*e*-TA) and *probabilistic*-TA (*p*-TA). Both approaches use the same concept of Pareto classification of slave processes, but they differ on the selection mechanism of the algorithms to be assigned to the slave processes.

5.1 Elitist TA

The *elitist*-TA (*e*-TA) is a natural extension of *s*-TA in which processors of similar characteristics are considered. In this approach, the master process selects randomly a slave process w_i belonging to the best front R_1 and another slave process w' corresponding to the worse front R_n. Next, the algorithm associated with w_i replaces the algorithm of *w'*. Thus, the algorithms with better performance are gradually replacing the algorithms with the worst performances. A single algorithm is replaced at each iteration. Algorithm 3 illustrates the sub-routine *Mapping* of the master process given in Algorithm 1.

Algorithm 3: Mapping for e-TA

Input: *P, H*, Ψ
Output: W

```
R = Pareto Classification of results from Slaves (Ψ )
if |R| > 1 then
   Randomly select a slave process w' ∈ $R_1$
   Randomly select a slave process $w_i$∈ $R_n$
   Replace algorithm $h_{w'}$ with $h_{wi}$
end if
return W;
```

5.2 Probabilistic TA

Clearly, the *e-TA* approach presents a high elitist pressure on the method to select algorithms for a new iteration. Furthermore, an algorithm replaced loses any chance of being selected in a future iteration. This suggests that *e*-TA can have stagnation with some algorithm whose initial performance was good. However, this algorithm might generally be bad in the long run. Clearly, this could provoke a bad performance of the TA with some problems. In order to prevent the loss of diversity of algorithms in slave processes, the following approach is proposed. Algorithm selection is based on a probability distribution based on the classification of slaves processes *R*. This approach is called *probabilistic*-TA (*p*-TA) and the probability of selecting a slave process can be calculated according to equation (3).

$$p(R_l) = \frac{2 \cdot (|R| - l + 2)}{(|R| + 1) \cdot (|R| + 2)} \quad \text{con } l=1, 2, \ldots, |R|+1 \tag{3}$$

All slave processes belonging to a given classification R_l (l = 1, 2,..., |R|) receive the same selection probability. In this approach, algorithms unallocated to any slave process are classified as $R_{|R|+1}$. This will ensure that any algorithm has a probability of selection greater than zero in any iteration, regardless of their performance in past iterations. Considering equation (3), note that: $p(R_1) > p(R_2) > \ldots > p(R_{|R|+1})$ and $\sum_{l=1}^{|R|+1} p(R_l) = 1$. The outline of *p*-TA is presented in Algorithm 4.

Algorithm 4: Mapping for *p-TA*

Input: *P, H, Ψ*
Output: *W*

```
$R$ = Pareto Classification of results from Slaves ( $\Psi$)
Calculates $p(R_l)$ using equation (3)
for ($i$ = 1 to $|W|$) do
   $w'$= Select a Slave Process of $R$ according $p(R_l)$
   Replace algorithm $h_{wi}$ with $h_{w'}$
end for
return W;
```

6 Experimental Analysis

Experiments carry out show the behavior of the proposed approaches *e-TA* and *p-TA* when applied to different combinational problems. The following set of bi-objective problems have been selected for this paper: *kroab100* and *kroac100*, instances of TSP, *qapUni.75.0.1* and *qapUni.75.p75.1*, instances of QAP, and *c101* and *rc101*, instances of VRPTW [6]. The set of MOACO algorithms considered were H = {BI-ANT, BIMC, COMP, MOACS, M3AS, MOAQ, MOA, PACO, MAS}, mentioned in Table 2 of Section 4. For comparison reasons, the parameters used in the MOACO algorithms, presented in Table 3, were chosen very similar to those used in [6]. The TAs were implemented in C + + (GNU GCC Compiler) and PVM library 3.4.5 [11], on a Linux operating system, Fedora 4 distribution. The experiments were performed on a network of 10 homogeneous workstations (9 slave computers and a master computer), 2 GHz processor and 512 MB RAM, i.e., $P = \{p_1, p_2, \ldots, p_9 \mid p_i \equiv p_j \; \forall \text{i, j}\}$.

Table 3. Configuration parameters of MOACOs

Parameter	*Value*
Number of ants	10
Relative influence between pheromones and visibility	0.5
Learning factor (*MOAQ*)	0.8
Optimality policy (*MOAQ*)	0.3
Evaporation rate	0.1
Initial level of pheromone	1
Omicron factor (*MOA*)	10
Re-initiation factor (*MAS*)	500
Exploration vs. Exploitation Probability (*MOACS*)	0.5

For the Pareto classification of slave processes (R), this work only considers two performance figures: M_3 and M_5, i.e. $|N_k|$=2. Since *s*-TA considers only a single metric ($|N_k|$=1), it uses M_5 according to the original proposal [6]. In order to experimentally verify performance of the proposed approaches, this work also implemented parallel versions of MOACS (*p*MOACS) and M3AS (*p*M3AS). These algorithms were selected considering their excellent experimental results reported in [3, 6, 17 and 18].

6.1 Experimental Design

Let's A = {*s-TA, e-TA, p-TA, pMOACS, pM3AS*} be a set of approaches to be compared in the experimental tests. The following steps outline the experimental tests applied to each instance:

1. 10 executions were carried out for each approach.
2. A known Pareto front F_{known} was calculated considering all Pareto fronts.
3. Each calculated Pareto front was compared to F_{known} obtaining several performance measures presented in Table 1.

To report results, an execution time of 1000 seconds was experimentally adopted as stop criterion, even though other criteria were also tested.

The above indicates that in overall, the experimental test consists in 300 run (5 algorithms • 3 problem • 2 instances • 10 times), i.e. more than *83 hours* of run time using a network of 10 computers.

6.2 General Results

In order to have a clear appreciation of the performance of the different approaches, this sub-section provides the overall averages for each test instance as well as a ranking of all performance figures considered in this work.

Table 4 shows the normalized *general averages* considering all 10 runs (with 1 as the best value and 0 as the worst case), as well as *standard deviation* (σ) associated with each instance and each performance figure obtained after the completion of the proposed test. It can be noted that for *kroab100* the best partial average was obtained by the *p-TA* with 0.96 while *s-TA* had a poor performance with only 0.41, lower than the one obtained by the *pM3AS* with a score of 0.47. The proposed *e-TA* also had a remarkable performance with an average of 0.86, well above to the rest of the approaches.

Table 4. Results for each bi-objective instance, using a master and 9 slave computers

Instance	*Algorithms*	M_1	M_2	M_3	M_4	M_5	σ	*Average*
kroab 100	*p-TA*	0.97	0.95	0.87	1	1	0.05	0.96
	e-TA	1	1	0.87	0.69	0.76	0.14	0.86
	s-TA	0.1	0.79	0.99	0.02	0.13	0.45	0.41
	pMOACS	0	0.93	1	0	0.14	0.51	0.41
	pM3AS	0.43	0.83	0.90	0.04	0.16	0.39	0.47
kroac 100	*p-TA*	0.54	1	0.91	0.49	0.50	0.25	0.69
	e-TA	1	1	0.89	1	1	0.05	0.98
	s-TA	0	0.82	0.96	0	0.03	0.48	0.36
	pMOACS	0.37	0.91	0.95	0.16	0.17	0.30	0.51
	pM3AS	0.07	0.83	1	0.10	0.11	0.45	0.42
qapUni 75.0.	*p-TA*	0.70	0.50	0.74	0.86	0.88	0.15	0.74
	e-TA	1	0.53	1	1	1	0.21	0.91
	s-TA	0.20	0.61	0.79	0.08	0.25	0.30	0.39
	pMOACS	0.59	0.52	0.88	0.47	0.50	0.17	0.59
	pM3AS	0	1	0.91	0	0.25	0.49	0.43
qapUni 75.p75.1	*p-TA*	1	0	0	1	1	0.55	0.60
	e-TA	0	1	1	0	0	0.55	0.40
	s-TA	0.27	0.33	0.8	0	0	0.33	0.28
	pMOACS	0.09	0.67	0.37	0	0	0.29	0.23
	pM3AS	0	0	0	0	0	0	0
c101	*p-TA*	1	-	-	1	1	0.55	0.60
	e-TA	0	-	-	0	0	0	0
	s-TA	1	-	-	1	1	0.55	0.60
	pMOACS	1	-	-	1	1	0.55	0.60
	pM3AS	0.64	-	-	0	0	0.29	0.13
rc101	*p-TA*	1	0	0	1	1	0.55	0.60
	e-TA	0.26	0	0	0	0	0.12	0.05
	s-TA	0.69	0	0	0	0	0.31	0.14
	pMOACS	0	1	1	0	0	0.55	0.40
	pM3AS	0.90	0	0	0	0	0.40	0.18

With regard to *kroac100*, *e-TA* obtained a remarkable average of 0.98. Here, *s-TA* was again the one with the worst performance, with just 0.36. Note that *pMOACS* obtained a good performance reaching an average of 0.51 whereas *p-TA* obtained the second best average performance of 0.69. Considering the two instances of the QAP, the proposed TA approaches are clearly superior.

The VRPTW instances used for this work are characterized by having a Pareto optimal front with a single solution, i.e. $|F_{known}| = 1$. Therefore, the distribution and extension figures are meaningless. For the instance, *p-TA*, *s-TA* and *pMOACS* obtained the same performance average with *c101*. This is because the previous approaches managed to find the only solution belonging to F_{knowm} in all 10 executions. It can be observed that in *rc101*, *p-TA* has achieved a better performance having obtained an average of 0.60, whereas *e-TA* obtained the worse behavior with an average of 0.14. On the other hand, *pMOACS* obtained the second best performance with an average of 0.40.

Table 5 presents the obtained ranking considering the average reached considering all the execution in all instances of the test problem. A clear robustness of *p-TA* in comparison with all other approaches can be observed. This is consistent with a balance reached on the various performance figures. Note that for the Distance (M_1), the Error (M_4) and the ONVGR (M_5), *p-TA* has obtained significant performance values.

Regarding the Extension (M_3) and Distribution (M_2), the best values correspond to *pMOACS*. It should be noted that *e-TA* achieved the second best average with 0.53 but with a standard deviation of only 0.08. On the other hand, *pMOACS* achieved an average value higher than that obtained by the *s-TA*, which confirms the goodness of MOACS [3].

Table 5. Global Average

Algorithms	*M_1*	*M_2*	*M_3*	*M_4*	*M_5*	*σ*	*Global Average*
p-TA	0.87	0.41	0.42	0.89	0.90	0.26	0.70
e-TA	0.54	0.59	0.63	0.45	0.46	0.08	0.53
s-TA	0.38	0.43	0.59	0.18	0.24	0.16	0.36
pMOACS	0.34	0.67	0.70	0.27	0.30	0.21	0.46
pM3AS	0.34	0.44	0.47	0.02	0.09	0.21	0.27

7 Conclusions

This paper proposes two new TA approaches: the *elitist*-TA (*e-TA*) and the *probabilistic*-TA (*p-TA*). Basically, both approaches classify the algorithms of a TA according to performance figures of the Pareto solutions they calculate. Experimental results indicate that the new proposals are promising approaches since they achieved greater robustness considering different types of combinational problems. Specifically, *p-TA* has succeeded in demonstrating a better balance between the various figures of merit, which define the quality of solutions generated by a multi-objective algorithm. This robustness is justified due to the combination of available algorithms, being these algorithms classified and selected fairly from the viewpoint of a multi-objective Pareto context.

As future work, it is intended to conduct more experimental tests on more test instances as well as real engineering problems, looking to refine the proposed approach toward becoming "smarter" methods to combine different algorithms in a distributed system.

References

1. Coello, C., Lamont, G., Van Veldhuizen, D.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, Heidelberg (2007)
2. Iredi, S., Merkle, D., Middendorf, M.: Bi-Criterion Optimization with Multi Colony Ant Algorithms. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 359–372. Springer, Heidelberg (2001)
3. García-Martínez, C., Cordón, O., Herrera, F.: An Empirical Análisis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP. In: ANTS Workshop, pp. 61–72 (2004)
4. Moore, J., Chapman, R.: Application of Particle Swarm to Multiobjective Optimization. Department of Computer Science and Software Engineering, Auburn University (1999)
5. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
6. Paciello, J., Martínez, H., Barán, B.: Team Algorithms for Ant Colony based Multi-objective Problems. In: ASAI 2006, Mendoza, Argentina (September 2006)
7. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms. Empirical result. Evolutionary computation 8(2), 173–195 (2000)
8. Van Veldhuizen, D.A.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University (June 1999)
9. Barán, B., Kaszkurewicz, E., Bhaya, A.: Parallel Asynchronous Team Algorithms: Convergence and Performance Análisis. IEEE Transactions on Parallel & Distributed Systems 7(7), 677–688 (1996)
10. Fernandez, J., Barán, B.: Multiobjective Evolutionary Elitist Team Algorithm. In: XXXI Conference Latino-American of Informatics, CLEI, Cali, Colombia (2005)
11. Geist, A., et al.: PVM: Parallel Virtual machine - A user's guide and Tutorial for Networked parallel Computing. MIT Press, Cambridge (1994)
12. Cantu-Paz, E.: Designing efficient and accurate parallel genetic algorithms., Technical Report 2108, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois (1999)
13. Stützle, T.: Parallelization strategies for ant colony optimization. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 722–731. Springer, Heidelberg (1998)
14. Iredi, S., Merkle, D., Middendorf, M.: Bi-Criterion Optimization with Multi Colony Ant Algorithms. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 359–372. Springer, Heidelberg (2001)
15. Doerner, K., Gutjahr, W., Hartl, R., Strauss, C.: Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection. In: Proceedings of the 4th Metaheuristics International Conference, Porto, pp. 243–248 (2001)
16. Doerner, K., Hartl, R., Reimann, M.: Are COMPETants more competent for problem solving? – the case of a multiple objective transportation problem. Central European Journal of Operations Research 11(2), 115–141 (2003)
17. Schaerer, M., Barán, B.: A multiobjective Ant Colony System for Vehicle Routing Problems with Time Windows. In: Proc. Twenty first IASTED International Conference on Applied Informatics, Insbruck, Austria, pp. 97–102 (2003)
18. Pinto, D., Barán, B.: Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimization approach. In: II IFIP/ACM Latin-American Networking Conference, Cali, Colombia (October 2005)

19. Mariano, C., Morales, E.: A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks. Technical Report HC-9904, Mexican Institute of Technology of water, Mexico (June 1999)
20. Gardel, P., Barán, B., Estigarribia, H., Fernandez, U.: Applications of Omicrom ACO to Reactive Compensation Problem in Multiobjective context. Argentine Congress of Computer Science. Concordia – Argentina (2005)
21. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India (2000)

Multiobjective Optimization on a Limited Budget of Evaluations Using Model-Assisted $\mathcal{S}$-Metric Selection

Wolfgang Ponweiser[1], Tobias Wagner[2], Dirk Biermann[2], and Markus Vincze[1]

[1] Automation and Control Institute
Vienna University of Technology, 1040 Vienna, Austria
{ponweiser,vincze}@acin.tuwien.ac.at
[2] Institute of Machining Technology (ISF)
Technische Universität, 44227 Dortmund, Germany
{wagner,biermann}@isf.de

Abstract. Real-world optimization problems often require the consideration of multiple contradicting objectives. These multiobjective problems are even more challenging when facing a limited budget of evaluations due to expensive experiments or simulations. In these cases, a specific class of multiobjective optimization algorithms (MOOA) has to be applied. This paper provides a review of contemporary multiobjective approaches based on the singleobjective meta-model-assisted 'Efficient Global Optimization' (EGO) procedure and describes their main concepts. Additionally, a new EGO-based MOOA is introduced, which utilizes the $\mathcal{S}$-metric or hypervolume contribution to decide which solution is evaluated next. A benchmark on recently proposed test functions is performed allowing a budget of 130 evaluations. The results point out that the maximization of the hypervolume contribution within a real multiobjective optimization is superior to straightforward adaptations of EGO making our new approach capable of approximating the Pareto front of common problems within the allowed budget of evaluations.

Keywords: Efficient Global Optimization, $\mathcal{S}$-metric, Design and Analysis of Computer Experiments, Multiobjective Optimization, Real-World Problems.

1 Introduction

Modern industrial processes become more and more complex. They consist of multiple stages, each configurable by several parameters. Thus, the determination of adequate parameter settings is a recurring task. It is especially challenging in case of expensive experiments or numerical computer simulations, where every realization involves high personnel or material expenses or requires immense calculation time. In these cases it is essential to obtain the desired outcome within a small number of evaluations.

This task becomes even more challenging in case of multiple, potentially contradicting objectives, such as product quality and cost. For these multiobjective optimization problems (MOP), the target is to find Pareto-optimal solutions, i.e., solutions where an objective cannot be improved without deteriorating at least one other. The challenges involved in solving a MOP are to converge towards Pareto-optimal solutions and generate a well distributed solution set, which covers the entire Pareto front [1].

In order to apply MOOA to most real-world problems, these challenges have to be mastered efficiently within a minimum number of objective evaluations [2,3]. Thus, in this paper we present a review of recent MOOA for these problems and introduce a new enhanced approach called '$\mathcal{S}$-Metric-Selection-based Efficient Global Optimization' (SMS-EGO). In particular, two state-of-the-art approaches – Knowles' ParEGO [2] and a MOOA presented by Jeong and Obayashi [4] – are reviewed and compared to SMS-EGO. All these methods compensate the limited amount of information by using approximate meta-models of the objective functions, on which a comprehensive optimization is performed to determine the next solution for evaluation on the actual problem. Since the actual objective evaluations are expensive, only a few iterations of the MOOA can be performed. Thus, the runtime of the optimization approach itself is not a critical issue. The restriction to a limited amount of evaluations is motivation as well as prerequisite.

In the next section, the state of the art in meta-model-assisted multiobjective optimization is presented. The EGO approach, which represents one of the most famous meta-model-based singleobjective optimization algorithms, is described in section 3. This approach allows solving common singleobjective problems with up to six dimensions on a budget of about 100 evaluations [5]. Subsequently, recent concepts to transfer EGO to multiple objectives are presented, and the new SMS-EGO is introduced. The implementation of the evaluated algorithms, the benchmark test functions, which are used to analyze whether the performance of EGO can be transferred to MOP, as well as the experimental results are described in section 4. Finally, the findings are summarized in section 5.

2 State-of-the-Art

A common approach to solve optimization tasks in case of expensive evaluations is to introduce an intermediate modeling step [6]. In this study, we focus on 'Design and Analysis of Computer Experiments' (DACE) [7]. DACE is a famous meta-modeling approach, which utilizes the assumption of close solutions being more likely to have similar objective values. This approach is widely accepted as a meta-model for deterministic non-linear functions. Furthermore, a recent study [3] supports its application also to problems with noisy evaluations. In the following we denote these kinds of meta-models as DACE models or DACE approximation.

Most papers on DACE-model-assisted MOOA are purely application oriented [8,9]. These algorithms simply apply the approximation as a surrogate function to reduce expensive evaluations. No comparable benchmark results are available. Furthermore, some MOOA additionally use the uncertainty of predictions to balance between local and global search. Emmerich et al. [10,11,12] use local DACE models to determine lower and upper confidence bounds for a prescreening of solutions within classical multiobjective evolutionary algorithms (MOEA, EMOA), such as NSGA-II, ε-MOEA, and SMS-EMOA. They report successful results on test functions and real-world problems. However, the confidence bounds are only used to evaluate solutions which are generated within the evolution. No criterion for a specific optimization has been developed.

Recent approaches [2,4,13] transfer concepts of the popular EGO approach to multiple objectives and are described and reviewed in the next sections.

3 Efficient Global Optimization

The EGO approach by Jones et al. [5] is the most famous DACE-model-based optimization algorithm. The prediction of DACE models is based on the n already evaluated solutions $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ by modeling the corresponding error $\epsilon(\mathbf{x})$ to a constant regression model $y(\mathbf{x}) = \mu + \epsilon(\mathbf{x})$. In this μ denotes the mean of the observations and $\epsilon(\mathbf{x})$ is assumed to have a mean of zero and covariance $Cov(\epsilon(\mathbf{x}), \epsilon(\mathbf{x}^{(j)})) = \sigma^2 R$ between $\epsilon(\mathbf{x})$ and each observed error $\epsilon(\mathbf{x}^{(j)})$ according to the measured process variance σ^2 and the correlation model $R(\mathbf{x}, \mathbf{x}^{(j)}, \theta, \mathbf{p}) = \prod_{i=1}^{d} \exp(-\theta_i |x_i - x_i^{(j)}|^{p_i}),\ j = 1, \ldots, n$, with d being the dimension in decision space. The modeling parameter $p_i \in]0, 2]$ controls the smoothness of the approximated function, and the parameter $\theta_i > 0$ specifies the activity in dimension i. Both are determined via maximum likelihood estimation.

A key feature of DACE models is the approximation of the corresponding uncertainty of a prediction. Using this information, the 'Expected Improvement' (EI) measures the expected value of improvement compared to the currently found minimum f_{min}. It can be calculated based on the predicted function value $\hat{y}$ and standard deviation $\hat{s}$

$$E[I(x)] = (f_{min} - \hat{y})\Phi(u) + \hat{s}\phi(u),\ u = \frac{f_{min} - \hat{y}}{\hat{s}}, \quad (1)$$

where Φ and ϕ denote the normal cumulative distribution function and the normal probability distribution function, respectively. In EGO the maximization of the EI provides the 'infill sampling criterion' [14] to determine the next point for evaluation. Due to the unknown correlation parameters and the resultant undervaluation of $\hat{s}$, this approach may lead to an undesired local search. However, approaches have been developed to overcome this drawback [14,15].

3.1 Adaptations of EGO for Multiobjective Problems

In case of m concurrent objectives, a separate model can be built for each objective dimension. Thereby, vectors of predicted values and estimated uncertainties are available. The calculation of the EI with respect to multiple objectives has been formerly derived by Keane [13]. He avoids the problem of selecting an appropriate reference vector $\mathbf{f}_{min}$ (cf. eq. 1) by computing the probability of augmenting the current Pareto front or dominating at least one of its solutions. Afterwards, the centroid of the distribution of the corresponding probability density can be used as expected solution to calculate an indicator of improvement to the current Pareto front. For $m > 2$, both tasks incorporate a numerically demanding partitioning of the objective space, for which no free implementations are available. Thus, this approach could not be benchmarked within our study. Additionally, two straightforward solutions to this problem have been published, which are described in the following subsections.

Multiobjective EI Optimization. Jeong and Obayashi [4] present an approach which directly uses the EI in each objective separately as fitness vector in a multiobjective optimization. For each objective they generate a DACE model and determine the best solution found at that time. Subsequently, the EI of a solution can be calculated for an optimization based on MOEA. Since MOEA usually obtain large sets of solutions, a small, yet representative sample of the population has to be obtained. To accomplish this, Jeong and Obayashi choose the m solutions having the highest EI values on each separate DACE model. Additionally, they keep the solution located closest to the center of the area, which is spanned by the final MOEA population in objective space.

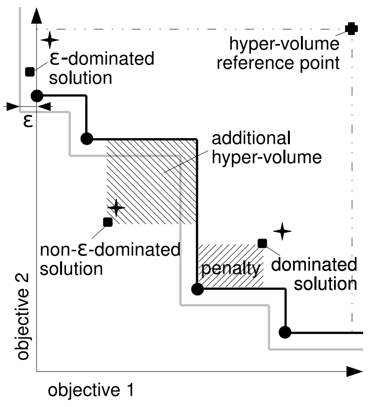
Multiple Singleobjective EI Sampling. ParEGO [2] developed by Knowles reverses the processing steps of model building and objective integration. It obviates the necessity of considering a multiobjective EI by first reducing the MOP to a singleobjective problem via an augmented Tchebycheff aggregation. In order to find solutions covering the whole Pareto front, the corresponding weight vector is randomly changed per iteration. The possible weight vectors are a priori calculated and evenly distributed. After combining the objectives, the scheme of EGO can be applied accordingly. Knowles seems to be the only author, who provides comparative results on established test problems with respect to a limited amount of evaluations [2,16]. He successfully compared ParEGO with random search, NSGA-II [17] as well as a binary-search-tree-based low-budget variant of MSOPS [16]. Nevertheless, a comparison to algorithms, which also use DACE meta-models, has not been performed.

$\mathcal{S}$-Metric Selection-based Efficient Global Optimization (SMS-EGO). The fundamental target of any MOEA consists in the improvement of the internal Pareto front approximation. Emmerich et al. [12] provide techniques to use common MOEA selection principles based on vectors of predicted values $\hat{\mathbf{y}}$ and estimated uncertainties $\hat{\mathbf{s}}$. They report best results using the lower confidence bound (LCB) $\hat{\mathbf{y}}_{pot} = \hat{\mathbf{y}} - \alpha\hat{\mathbf{s}}$ for a given confidence level $p_\alpha = (1 - 2\Phi(\alpha))^m$, which follows the non-error principle [10] to avoid the non-consideration of potentially promising solutions.

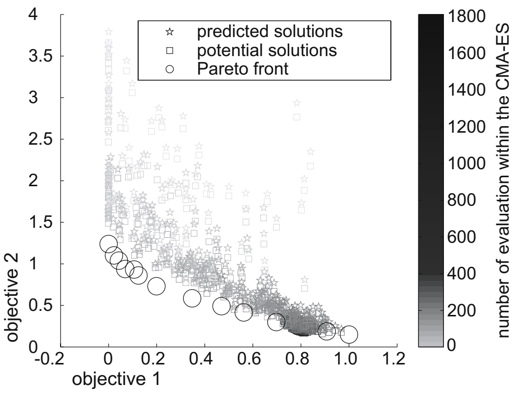
In our SMS-EGO approach, the idea of Emmerich et al. [11] to calculate the $\mathcal{S}$-metric [18] or hypervolume contribution of $\hat{\mathbf{y}}_{pot}$ to the current Pareto front approximation is extended to an independent infill criterion. The $\mathcal{S}$-metric contribution is chosen since it requires no normalization of the objective space [19] and holds some desired theoretical properties [20]. Wagner et al. [21] showed that a selection routine based on this contribution is superior to popular MOEA and also scales well with the number of objectives. As aforementioned, the problem of its computational complexity can be disregarded for the class of problems focused on.

Due to the use of the LCB, potential solutions can be predicted slightly beyond the real objective space. In order to tackle this problem as well as to support a good distribution, additive ε-dominance [20] is applied. The assignment of the vector ε is managed by introducing an adaptive scheme, which aims on a maximum number of individuals in the Pareto front approximation $\varepsilon = \frac{\Delta\Lambda}{|\mathbf{\Lambda}|} + c \cdot n_{left}$, $\Delta\Lambda = \max(\mathbf{\Lambda}) - \min(\mathbf{\Lambda})$, where $\mathbf{\Lambda}$ refers to the current Pareto front approximation, n_{left} denotes the number of remaining evaluations, and $c = 1 - 1/(2^m)$ is a correction factor, which constitutes the idealized probability for a remaining solution of being non-dominated [21].

In the calculation of the internal fitness value, three cases are considered. In the first case of a non-ε-dominated solution, its contribution to the $\mathcal{S}$-metric is calculated by



(a)



(b)

Fig. 1. (a) Graphical explanation of the evaluation of solutions within SMS-EGO. (b) An exemplary run of the model-based internal optimization on the ZDT1 test function. The grayscales express the sequence of evaluations performed by the CMA-ES. Consequently, a potential solution filling the gap in the Pareto front around $(0.8, 0.2)^T$ has been found.

$f = \mathcal{S}(\mathbf{\Lambda}) - \mathcal{S}(\hat{\mathbf{y}}_{pot} \cap \mathbf{\Lambda})$. The reference point needed for this calculation is defined by $\max(\mathbf{\Lambda}) + \mathbf{1}$ according to Emmerich et al. [22]. Second, in the case of a dominated solution, a penalty p is added for each dominating point. To keep the penalty close to the $\mathcal{S}$-metric, the differences in each of the m objective dimensions are multiplied. Additionally, a slight transformation is performed to assure that a positive penalty is assigned to weakly dominated solutions

$$p = \sum_{\mathbf{y}^{(i)} \in \mathbf{\Lambda}} \begin{cases} -1 + \prod_{j=1}^{m} \left(1 + (\hat{y}_{pot,j} - y_j^{(i)})\right) & \mathbf{y}^{(i)} \preceq \hat{\mathbf{y}}_{pot} \\ 0 & \text{otherwise} \end{cases} . \qquad (2)$$

In the third case of ε-dominated solutions, which are not dominated in the strict sense, the penalty is limited to the objectives that are inferior with respect to the considered Pareto front solution.

All three cases are visualized in Fig. 1 for $m = 2$. The preferred values of this fitness function are negative since new, non-dominated solutions increase the hypervolume of the current Pareto front approximation. In areas of dominated solutions, the search is guided towards non-dominated solutions by means of the penalty term. This measure can be minimized by any global singleobjective optimizer. Corresponding to the EGO procedure, only the solution achieving the minimum of this criterion is selected to be evaluated on the actual problem. Afterwards, the models are updated based on the new observation in order to utilize as much information as possible.

4 Experiments

In this section a comprehensive analysis of the results, which can be achieved on a budget of 130 function evaluations, is provided. To accomplish this, the approaches of

Jeong and Obayashi [4], Knowles' ParEGO [2], and the newly proposed SMS-EGO are benchmarked on established test functions.

Pre-experimental planning: Based on recent suggestions for performance assessment [23], five test functions are selected. More precisely, R_ZDT1 (biobjective, unimodal), R_ZDT4 (biobjective, multimodal), and two R_DTLZ2 variants with three and five objectives (scalability) [24] are chosen. The decision space dimension d is decreased to six in order to facilitate the modeling and to accord with typical numbers of process parameters in real-world processes [3]. Furthermore, the domain of R_ZDT4 has been reduced to three variables and $x_2, x_3 \in [-1, 1]$ to obtain a manageable number of only 20 local optima. This relaxed version is denoted as R_ZDT4$_{relax}$. Additionally, the OKA2 [25] test function is considered since it provides a challenging Pareto set in terms of shape and distribution.

Setup: All algorithms are implemented in MATLAB®. As suggested by Knowles, they start with an initial sampling of $n_{init} = 11d - 1$ solutions based on a Latin hypercube design (LHD) within the given box constraints [2]. This kind of random design is appropriate for real-world applications since in most cases the interesting parameter region is identified in a screening and then sampled by some kind of space-filling design. Furthermore, LHD have been proven to be suited for the generation of DACE models [5,7]. The design is evaluated, and the parameters of the DACE models are calculated by maximum likelihood estimation using Hansen's CMA-ES [26] implementation[1]. Whenever the CMA-ES is applied, the default values are chosen for all parameters, and three stopping critera, i.e., a maximum number of $4000d$ evaluations and the convergence of the population in the objective or decision space, are set up. For each test function, an amount of 130 evaluations on the actual test function is allowed. To speed up the Pareto front calculation, external C-code programed by Yi Cao is applied[2].

Jeong's approach is implemented using code of NSGA-II[3] for the internal multiobjective EI optimization. According to Jeong's suggestions, the population size and the number of generations are set to 512 and 100, respectively [4]. The center solution is determined as minimizer of the uniformly weighted augmented Tchebycheff aggregation of the normalized Pareto front solutions. Also in ParEGO, the augmented Tchebycheff aggregation is implemented according to Knowles using $\rho = 0.05$ and normalized objective values [2]. Within SMS-EGO the $\mathcal{S}$-metric calculation is performed by a C implementation of Fonseca et al. [27]. The factor of the estimated uncertainty is set to $\alpha = \Phi^{-1}(0.5 + \frac{1}{2^m})$. The infill criteria of ParEGO and SMS-EGO are maximized using the CMA-ES.

The PISA test environment is applied for the evaluation of the results. As performance measures, the unary hypervolume indicator [18], Hansen and Jaszkiewicz's R_2 indicator [28], and the unary epsilon indicator [20] are used. These indicators are suggested for multiobjective performance assessment and evaluate both, convergence and distribution [20]. Additionally, the mean distance of the approximated Pareto front to the real one is calculated analytically on R_DTLZ2 to allow the separated consideration of the convergence. For each algorithm and each test function, five runs are performed.

[1] http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html

[2] http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=17251

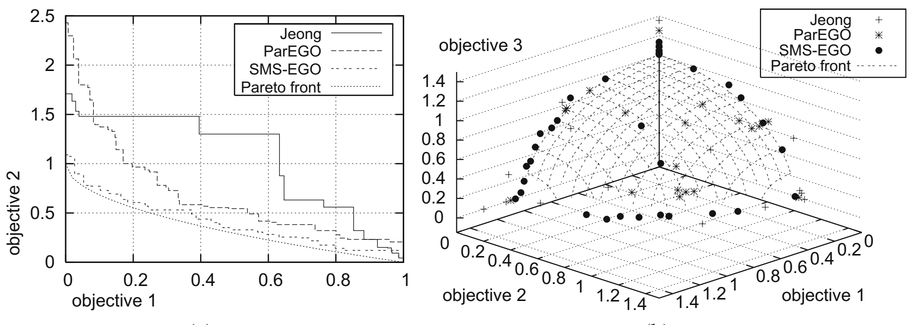
[3] http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10429

Table 1. The table shows the median and worst-case results of the hypervolume, R_2, and ε-indicator on OKA2, R_ZDT1, R_ZDT4$_{relax}$ and R_DTLZ2 (with three and five objectives). The smallest and highest value in each group are printed in bold and italics, respectively. The letters in brackets indicate whether the difference of the median value is statistically significant compared to Jeong (J), ParEGO (P), and SMS-EGO (S) based on a one-sided Kruskal Wallis test ($p < 0.01$).

test function	indicator	algorithm					
m / d		Jeong		ParEGO		SMS-EGO	
		median	max	median	max	median	max
OKA2	$\mathcal{S}$-metric	*3.41e-1* (-,-)	*4.01e-1*	2.47e-1 (-,-)	3.35e-1	**1.40e-1** (-,-)	**1.42e-1**
2 / 3	R_2	8.94e-2 (-,S)	1.37e-1	**2.42e-2** (J,S)	**7.00e-2**	*1.66e-1* (-,-)	*1.67e-1*
	ε	3.48e-1 (-,S)	4.07e-1	**2.87e-1** (J,S)	**3.59e-1**	*4.96e-1* (-,-)	*4.98e-1*
R_ZDT1	$\mathcal{S}$-metric	*2.60e-1* (-,-)	*2.75e-1*	1.21e-1 (J,-)	1.55e-1	**2.38e-2** (J,P)	**7.03e-2**
2 / 6	R_2	**3.48e-3** (-,-)	**5.82e-3**	*9.22e-3* (-,-)	1.50e-2	7.30e-3 (-,-)	*2.74e-2*
	ε	*3.23e-1* (-,-)	*3.60e-1*	1.49e-1 (J,-)	1.59e-1	**4.00e-2** (J,P)	**7.34e-2**
R_ZDT4$_{relax}$	$\mathcal{S}$-metric	*2.27e-1* (-,-)	*3.04e-1*	3.19e-1 (-,-)	3.68e-1	**7.58e-2** (J,P)	**1.16e-1**
2 / 3	R_2	**2.22e-2** (-,-)	4.42e-2	*4.87e-2* (-,-)	*1.33e-1*	2.25e-2 (-,-)	**4.24e-2**
	ε	2.27e-1 (-,-)	3.45e-1	*3.33e-1* (-,-)	*3.80e-1*	**1.04e-1** (J,P)	**1.41e-1**
R_DTLZ2	$\mathcal{S}$-metric	*9.31e-2* (-,-)	*1.09e-1*	6.79e-2 (J,-)	7.62e-2	**1.90e-2** (J,P)	**2.12e-2**
3 / 6	R_2	5.15e-5 (-,-)	5.36e-5	*6.11e-5* (-,-)	*7.63e-5*	**1.10e-5** (J,P)	**1.43e-5**
	ε	*1.97e-1* (-,-)	*2.03e-1*	1.58e-1 (J,-)	1.67e-1	**8.05e-2** (J,P)	**9.35e-2**
R_DTLZ2	$\mathcal{S}$-metric	2.60e-2 (-,-)	*1.36e-1*	*5.31e-2* (-,-)	8.26e-2	**1.22e-2** (J,P)	**1.76e-2**
5 / 6	R_2	2.31e-5 (-,-)	*5.19e-4*	*8.16e-5* (-,-)	1.14e-4	**5.68e-7** (-,-)	**9.93e-7**
	ε	1.69e-1 (-,-)	3.05e-1	*2.56e-1* (-,-)	*3.36e-1*	**1.44e-1** (J,P)	**1.66e-1**

Experimentation/Visualization: The results of the experiments are summarized in Table 1. The distribution of solutions within the median Pareto front approximation is exemplarily visualized in Fig. 2. In Fig. 3, boxplots of the mean distance to the Pareto front are shown for R_DTLZ2 with three and five objectives.

Observations: SMS-EGO performs significantly better than ParEGO and Jeong with respect to the ε- and hypervolume indicator on R_ZDT1, R_ZDT4$_{relax}$, and



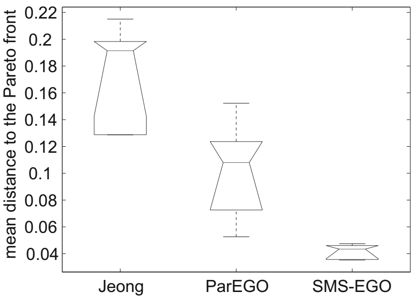
(a) (b)

Fig. 2. (a) The median attainment surfaces of all algorithms on ZDT1. (b) The distribution of solutions in the objective space of DTLZ2 with three objectives. Exemplarily, the runs are chosen which achieved the median result with respect to hypervolume indicator.

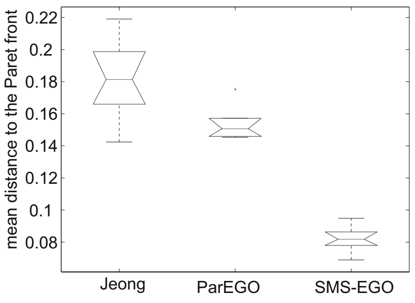
R_DTLZ2 with five objectives while being worse on OKA2 for the R_2 and ε measure. When three objectives are considered, SMS-EGO outperforms all other algorithms for all metrics. On this R_DTLZ2 instance, on R_ZDT1, and on OKA2, ParEGO also outperforms Jeong regarding the ε- and hypervolume indicator.

Discussion: Whereas the concepts of the hypervolume and the ε-indicator are directly considered within SMS-EGO, the R_2 metric and ParEGO are both based on augmented Tchebycheff aggregation. Thus, it is particularly surprising that SMS-EGO significantly outperforms ParEGO with respect to this metric on the threeobjective R_DTLZ2 test function. Furthermore, the results on the test functions, which feature more than two objectives, show that SMS-EGO copes best to increasing objective dimensions. This fact is visualized in the boxplots showing the distributions of the mean distance to the Pareto front on R_DTLZ2 with five objectives in Fig. 3 (b). The inferior results on OKA2 can be explained by the difficulty of this test function. SMS-EGO obtains one extremal solution with high accuracy ($\approx 10^{-9}$)). Other solutions are neglected since a comparable accuracy is necessary to provide further non-dominated solutions due to the steep ridges around the optimal area. Jeong and ParEGO are forced towards other solutions by their selection principles. Thus, they provide a better distribution, which results in significantly better indicator values. Nevertheless, they fail to provide close to optimal solutions. Consequently, OKA2-like problems with steep ridges cannot be solved within a reduced budget of just 130 evaluations using the proposed model-based approaches. The superior results of SMS-EGO on R_ZDT4$_{relax}$ indicate that the use of the LCB solution does not disregard global exploration compared to the EI used in ParEGO and Jeong.

In order to further analyze the distribution and the convergence behavior of the algorithms, Fig. 2 visualizes the median Pareto front approximations in the objective space and Fig. 3 shows boxplots of the mean distance to the Pareto front on both variants of R_DTLZ2. Jeong and Obayashi's approach covers only the boundaries of the Pareto front with a competitive accuracy, which leads to the worst distance values of all algorithms. This behavior may be caused by the dimension-based reference values for the



(a)



(b)

Fig. 3. Box plots of the mean distance to the Pareto front over five runs on DTLZ2 with three (a) and five objectives (b). The box extends from the lower quartile to the upper quartile, and a line is drawn at the median value. The spread of the sample is indicated by the whiskers. The notches represent a robust estimate of the uncertainty of the measured median for box to box comparison.

computation of the EI. ParEGO performs slightly better, but explores the boundaries of the Pareto front to a lesser extend. This problem of aggregation-based approaches using sets of weight vectors has already been observed by Wagner et al. [21]. SMS-EGO is the only MOOA that is able to approximately attain the complete Pareto fronts of R_ZDT1 and R_DTLZ2 with three and five objectives within the allowed budget of 130 evaluations, which is just slightly beyond the initial population size of most common MOEA (cf. Fig. 2 (a) and Fig. 3).

5 Conclusions

The optimization of most real-world problems requires an efficient use of evaluations. Thus, recent approaches, which transfer the singleobjective meta-model-assisted EGO approach to MOP, are presented and a new enhanced algorithm based on the $\mathcal{S}$-metric or hypervolume contribution (SMS-EGO) is introduced. A comprehensive benchmark is performed to analyze the results, which can be obtained on a budget of 130 evaluations. The SMS-EGO introduced in this paper performs significantly better on all considered R_ZDT and R_DTLZ2 instances. It is the only approach that is able to approximately attain the Pareto front of these problems under the given conditions while achieving both aims of multiobjective optimization, convergence and a good distribution of solutions.

The method of Keane [13] may show comparable results to SMS-EGO. However, his proposed improvement metric heavily depends on the scaling of the objectives. Since the use of the lower confidence bound for computing the hypervolume contribution in SMS-EGO also leads to some undesired side effects, such as the occurrence of dominated solutions, the implementation of his approach for formally determining the expected objective vector in arbitrary dimensions and the combination with the infill criterion of SMS-EGO are aspired.

Acknowledgments

This work was supported in part by the National Science Foundation under grant no. S9101-N13 and is based on investigations of the collaborative research center SFB/TR TRR 30, which is kindly supported by the DFG.

References

1. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Dissertation, Swiss Federal Institute of Technology (ETH), Zürich (1999)
2. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evolutionary Computation 10(1), 50–66 (2006)
3. Biermann, D., Weinert, K., Wagner, T.: Model-based optimization revisited: Towards real-world processes. In: [29], pp. 2980–2987
4. Jeong, S., Obayashi, S.: Efficient global optimization (EGO) for multi-objective problem and data mining. In: [30], pp. 2138–22145
5. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Global Optimization 13(4), 455–492 (1998)

6. Wang, G.G., Shan, S.: Review of metamodeling techniques in support of engineering design optimization. Journal of Mechanical Design 129(4), 370–380 (2007)
7. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–423 (1989)
8. Gonzalez, L.F., Walker, R., Periaux, K.S.,, J.: Multidisciplinary design optimisation of unmanned aerial systems (UAS) using meta model assisted evolutionary algorithms. In: 16th Australasian Fluid Mechanics Conf., pp. 471–474 (2007)
9. D'Angelo, S., Minisci, E.: Multi-objective evolutionary optimization of subsonic airfoils by kriging approximation and evolution control. In: [30], pp. 1262–1267
10. Emmerich, M., Naujoks, B.: Metamodel-assisted multiobjective optimisation strategies and their application in airfoil design. In: Parmee, I.C. (ed.) Adaptive Computing in Design and Manufacture VI, pp. 249–260. Springer, Heidelberg (2004)
11. Naujoks, B., Beume, N., Emmerich, M.: Metamodel-assisted SMS-EMOA applied to airfoil optimisation tasks. In: Schilling, R., et al. (eds.) Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2005) (CD-ROM). FLM (2005)
12. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels. IEEE Trans. Evolutionary Computation 10(4), 421–439 (2006)
13. Keane, A.J.: Statistical improvement criteria for use in multiobjective design optimization. AIAA 44, 879–891 (2006)
14. Sasena, M., Papalambros, P., Goovaerts, P.: Exploration of metamodeling sampling criteria for constrained global optimization. Engineering Optimization 34, 263–278 (2002)
15. Ponweiser, W., Wagner, T.: Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In: [29], pp. 3514–3521
16. Knowles, J.D., Hughes, E.J.: Multiobjective optimization on a budget of 250 evaluations. In: [31], pp. 176–190
17. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: Schoenauer, M., et al. (eds.) Proc. 6th Int'l Conf. Parallel Problem Solving from Nature, pp. 849–858. Springer, Heidelberg (2000)
18. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms – A comparative case study. In: Eiben, A.E., et al. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
19. Knowles, J.: Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization. PhD thesis, Department of Computer Science, University of Reading (2002)
20. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. Trans. Evolutionary Computation 8(2), 117–132 (2003)
21. Wagner, T., Beume, N., Naujoks, B.: Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In: Obayashi, S., et al. (eds.) Proc. 4th Int'l Conf. Evolutionary Multi-Criterion Optimization, pp. 742–756. Springer, Heidelberg (2007)
22. Emmerich, M., Beume, N., Naujoks, B.: An emo algorithm using the hypervolume measure as selection criterion. In: [31], pp. 62–76 (2005)
23. Huang, V.L., Qin, A.K., Deb, K., Zitzler, E., Suganthan, P.N., Liang, J.J., Preuss, M., Huband, S.: Problem definitions for performance assessment of multi-objective optimization algorithms. Technical report, Nanyang Technological University, Singapore (2007)
24. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. MIT Evolutionary Computation 8(2), 173–195 (2000)

25. Okabe, T., Jin, Y., Olhofer, M., Sendhoff, B.: On test functions for evolutionary multi-objective optimization. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 792–802. Springer, Heidelberg (2004)
26. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
27. Fonseca, C., Paquete, L., Lopez-Ibanez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: Congr. Evolutionary Computation, pp. 1157–1163. IEEE, Los Alamitos (2006)
28. Hansen, M.P., Jaszkiewicz, A.: Evaluating the quality of approximations to the non-dominated set. Technical Report IMM1998-7, Technical University of Denmark (1998)
29. Michalewicz, Z., et al. (eds.): Proc. Congr. Evolutionary Computation. IEEE, Los Alamitos (2008)
30. Corne, D., et al. (eds.): Proc. Congr. Evolutionary Computation. IEEE, Los Alamitos (2005)
31. Coello, C.A.C., et al. (eds.): Proc. 3rd Int'l Conf. Evolutionary Multi-Criterion Optimization. Springer, Heidelberg (2005)

Approximating the Knee of an MOP with Stochastic Search Algorithms

Oliver Schütze[1], Marco Laumanns[2], and Carlos A. Coello Coello[1]

[1] CINVESTAV-IPN, Computer Science Department, Mexico City, Mexico
{schuetze,ccoello}@cs.cinvestav.mx
[2] ETH Zurich, Institute for Operations Research, Zurich, Switzerland
laumanns@ifor.math.ethz.ch

Abstract. In this paper we address the problem of approximating the 'knee' of a bi-objective optimization problem with stochastic search algorithms. Knees or entire knee-regions are of particular interest since such solutions are often preferred by the decision makers in many applications. Here we propose and investigate two update strategies which can be used in combination with stochastic multi-objective search algorithms (e.g., evolutionary algorithms) and aim for the computation of the knee and the knee-region, respectively. Finally, we demonstrate the applicability of the approach on two examples.

1 Introduction

In many real world problems several objective functions have to be optimized simultaneously. One typical goal for such *multi-objective optimization problems* (MOPs) is to identify the entire set of optimal solutions (the Pareto set) and its image in objective space, the Pareto front. However, since the Pareto set typically forms a (k-1)-dimensional object, where k denotes the number of objectives, this task may become too hard, in particular for more objectives. Instead, one can e.g. integrate the decision maker (DM) into the search process (e.g., with *interactive methods* [11]) or can compute selected points out of the Pareto set, which we address here. One such particular solution is the 'knee'[1] or the 'maximal bulge' of the Pareto front which is often preferred by many DMs since it represents for them the 'optimal compromise' in multi-objective optimization. In this paper we propose and investigate two archiving strategies for stochastic search algorithms which aim for the computation of such a knee and entire knee-regions (i.e., solutions where the bulge is maximal or nearly maximal) respectively. We consider here the bi-objective case (i.e., $k = 2$), but the results may be extended for larger number of objectives.

Knees or other related user preference areas in multi-objective optimization have been addressed in many works so far ([1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13]). For instance, in [1] a multi-objective evolutionary algorithm is presented which

[1] There exist different characterizations of the knee in literature which, however, lead to the same or to similar solutions in many cases.

focuses on the knee-regions of an MOP (using a different characterization of the knee). The approach which we propose here can be viewed as a possible alternative to this work. One advantage of our strategies is that they can easily be integrated into any given archiving strategy for a stochastic search procedure. In that case, the (additional) approximation of the knee comes for 'free' in the sense that no additional function call has to be spent.

2 Background

In the following we consider continuous multi-objective optimization problems

$$\min_{x\in Q}\{F(x)\}, \tag{MOP}$$

where $Q \subset \mathbb{R}^n$ is compact and F is defined as the vector of the objective functions $F : Q \to \mathbb{R}^k, \quad F(x) = (f_1(x), \ldots, f_k(x))$, and where each $f_i : Q \to \mathbb{R}$ is continuous.

Definition 1. *Let $v, w \in Q$. Then the vector v is* less than w *($v <_p w$), if $v_i < w_i$ for all $i \in \{1, \ldots, k\}$. The relation $\leq_p$ is defined analogously. $y \in Q$ is* dominated *by a point $x \in Q$ ($x \prec y$) with respect to (MOP) if $F(x) \leq_p F(y)$ and $F(x) \neq F(y)$. $x \in Q$ is called a* Pareto optimal point *or* Pareto point *if there is no $y \in Q$ which dominates x.*

The set of all Pareto optimal solutions is called the *Pareto set* (denoted by P_Q). The image of the Pareto set $F(P_Q)$ is called the *Pareto front*. Further, we need the following distances between different sets.

Definition 2. *Let $u \in \mathbb{R}^n$ and $A, B \subset \mathbb{R}^n$. The semi-distance $dist(\cdot,\cdot)$ and the* Hausdorff distance *$d_H(\cdot,\cdot)$ are defined as follows: $dist(u, A) := \inf_{v\in A} \|u - v\|$, $dist(B, A) := \sup_{u\in B} dist(u, A)$, and $d_H(A, B) := \max\{dist(A, B), dist(B, A)\}$.*

Finally, we need to define some straight lines in $\mathbb{R}^2$. For $y_1, y_2 \in \mathbb{R}^2$, $y_1 \neq y_2$, we define by $\mathcal{L}(y_1, y_2) := y_1 + \mathbb{R}(y_2 - y_1)$ the line which goes through y_1 and y_2.

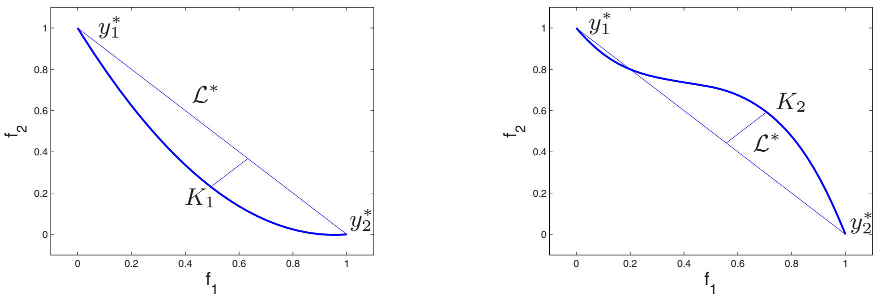
3 Characterization of the Knee

In this section we state one possible way to define the knee and modify it such that we can use it for our purpose.

According to [3], a knee of a Pareto curve is found by solving the following nonlinear programming problem (NLP):

$$\max_{p\in P_Q} dist(F(p), \mathcal{L}(F(p_1^*), F(p_2^*))), \tag{1}$$

where $p_i^* \in \arg\min_{x\in P_Q} f_i(x)$, $i = 1, 2$ (see also Figure 1). The knee as characterized by (1) can be interpreted as the maximal bulge of the curve with respect to the



(a) 'convex bulge' (b) 'concave bulge'

Fig. 1. Two 'knees' K_1, K_2 for different Pareto fronts as characterized by the maximal bulge of the Pareto curve with respect to $\mathcal{L}^* := \mathcal{L}(y_1^*, y_2^*)$

line $\mathcal{L}(F(p_1^*), F(p_2^*))$ which contains the two extreme points of the curve. We have chosen for this characterization since it requires no gradient information and is invariant to scalarization of the objectives.

Since we are interested in 'convex bulges' and not in 'concave bulges' which do intuitively not fit to the idea of minimization (see Figure 1, or [3]), we define the distance of the image $F(p)$ of a candidate solution to $\mathcal{L}(F(p_1^*), F(p_2^*))$ as follows:

$$D(p, p_1^*, p_2^*) := \begin{cases} dist(F(p), \mathcal{L}(F(p_1^*), F(p_2^*))) & \text{if } f_2(p) \le g(f_1(p)) \\ \text{-}\ dist(F(p), \mathcal{L}(F(p_1^*), F(p_2^*))) & \text{else} \end{cases}, \qquad (2)$$

where $g(x) = \mathcal{L}(F(p_1^*), F(p_2^*))$. Using this function and the fact that we are interested in convex bulges, we can modify NLP (1) by

$$\max_{x \in Q} D(x, p_1^*, p_2^*), \qquad (3)$$

which will be our 'knee' in the sequel.

4 The Algorithms

Here we propose two different update strategies for the approximation of a single knee as well as entire knee-regions and investigate the limit behavior of these algorithms.

First we are interested in obtaining *one* maximal bulge. Since in most cases (e.g., for all convex problems) 'the' knee is indeed unique it is sufficient to store one approximation — in addition to the approximations of the extreme points of the Pareto curve, since they are also not known a priori. Algorithm 1 shows one possible way to do this. The input parameters are the approximations m_1^0, m_2^0 of the extreme points, the current approximation K_0 of the knee as well as the new candidate solution $p \in Q$. Outputs are the new approximations of the extreme points (m_1, m_2) and of the knee (K). Theorem 1 shows that the maximal bulge (measured in objective space) is reached in the limit under certain assumptions and in the probabilistic sense.

Algorithm 1. $\{m_1, m_2, K\} := ArchiveUpdateMaxBulge1\ (p, K^0, m_1^0, m_2^0)$

```
1: if $f_1(p) < f_1(m_1^0)$ then
2:     $m_1 := p$
3: else
4:     $m_1 := m_1^0$
5: end if
6: if $f_2(p) < f_1(m_2^0)$ then
7:     $m_2 := p$
8: else
9:     $m_2 := m_2^0$
10: end if
11: if $D(p, m_1, m_2) > D(K^0, m_1, m_2)$ then
12:     $K := p$
13: else
14:     $K := K^0$
15: end if
```

Theorem 1. *Let (MOP) be given and $Q \subset \mathbb{R}^n$ be compact, let there be no weak Pareto points in $Q \backslash P_Q$, and $K^0, m_1^{(0)}, m_2^{(0)} \in Q$. Further, let $p_i^*, i = 1, 2$, as defined above with $F(p_1^*) \neq F(p_2^*)$, and*

$$\forall x \in Q \text{ and } \forall \delta > 0 : \qquad P\left(\exists l \in \mathbb{N} \ : \ p_l \in B_\delta(x) \cap Q\right) = 1, \tag{4}$$

where $B_\delta(x) := \{y \in \mathbb{R}^n \ : \ \|y - x\| < \delta\}$ and $P(A)$ denotes the probability for event A. Then, if Algorithm 1 is used to update the sequences $K_l, m_1^{(l)}, m_2^{(l)}, l \in \mathbb{N}$, it holds with probability one

(a)

$$m_1^{(l)} \to p_1^* \in \arg\min_{x \in P_Q} \ f_1(x) \quad \text{for } l \to \infty$$

$$m_{s_l}^{(l)} \to p_2^* \in \arg\min_{x \in P_Q} \ f_2(x) \quad \text{for } l \to \infty$$

(b)

$$D(K_l) \to \max_{x \in Q} D(x, p_1^*, p_2^*) \quad \text{for } l \to \infty.$$

Proof. (a) We prove the convergence of the sequence $(m_1^{(l)})_{l \in \mathbb{N}}$, the other statement is analogue. The claim follows, roughly speaking, by assumption (4) on the process to generate new candidate solutions and by the fact that the point with the smallest value according to f_1 which is found during the search is kept in the archive. To be more precise, let $x_1^* \in \arg\min_{x \in P_Q} f_1(x)$.

By (4) it follows that there exists for every $i \in \mathbb{N}$ with probability one a number j_i and a point $p_{j_i} \in B_{1/i}(x_1^*) \cap Q$. By construction of Alg. 1 it is $f_1(m_1^{(j_i)}) \leq f_1(p_{j_i})$. Thus, the claim follows since $p_{j_i} \to x_1^*$ for $i \to \infty$.

(b) The straight lines $\mathcal{L}_l(F(a_{m_1}^{(l)}), F(a_{m_2}^{(l)}))$ can be written as $g_l(x) = m_l x + b_l$. Let $a := f_1(p_1^*)$ and $b := f_1(p_2^*)$. Denote by $S_p = (x_p, y_p) \in \mathcal{L}_l$ the vector with minimal distance to the candidate solution p_l. It is easy to verify that $x_p \in [a, b]$ (see e.g. the Appendix). Thus, it is sufficient to consider the functions g_l on the interval $[a, b]$. Since $F(p_1^*) \neq F(p_2^*)$ and by part (a) of this theorem it follows that the g_l's are converging uniformly to $g = \mathcal{L}(F(p_1^*), F(p_2^*))$ on $[a, b]$, and thus we have with probability one

$$\max_{x \in Q} D(x, m_1^{(l)}, m_2^{(l)}) \to \max_{x \in Q} D(x, p_1^*, p_2^*), \quad \text{for } l \to \infty. \tag{5}$$

Let $p^* \in \arg\max_{x \in P_Q} D(x, p_1^*, p_2^*)$. By (4) it follows that there exists with probability one a subsequence of p_{j_i} of the candidate solutions such that $p_{j_i} \in B_{1/i}(p^*) \cap Q$. By construction of Alg. 1 it follows that $D(K_{j_i}, a_{m_1}^{(j_i)}, a_{m_2}^{(j_i)}) \geq D(p_{j_i}, a_{m_1}^{(j_i)}, a_{m_2}^{(j_i)})$. Using this and (5) we obtain with probability one

$$D(K_l, a_{m_1}^{(l)}, a_{m_2}^{(l)}) \to \max_{x \in Q} D(x, p_1^*, p_2^*),\ l \to \infty \tag{6}$$

and the proof is complete.

Next, we are interested to approximate beyond one knee solution the subset of the Pareto front where the bulge is 'large' since this entire set could be interesting for the decision maker ([1]). That is, for $M := \max_{x \in Q} D(x, p_1^*, p_2^*)$ and given a threshold $\Delta \in \mathbb{R}_+$ we are interested in the following set:

$$K_\Delta := \{x \in P_Q | D(x, p_1^*, p_2^*) \geq M - \Delta\} \tag{7}$$

Note that in case the knee is not unique all these points are included in K_Δ for every value of Δ, which is another motivation to approximate this set.

In Algorithm 2 we propose one possible archiving strategy which aims for the approximation of K_Δ. The notation is as in Alg. 1 with the difference that K is a set of points. In the following we investigate the limit behavior of the strategy under the same assumptions as above (Thm. 2). Before we can do this we need the following result.

Lemma 1. *Let $m_1, m_2, z, d \in Q$ with $f_1(m_1) < f_1(x) < f_1(m_2)$, and $g(f_1(x)) \leq f_2(x)$, where $g(\cdot) = \mathcal{L}_l(F(m_1), F(m_2))$, and $d \prec z$, and let m_1 and m_2 be mutually nondominating. Then $D(d, m_1, m_2) > D(z, m_1, m_2)$.*

Proof. Assume that $D(d, m_1, m_2) \leq D(z, m_1, m_2)$. Let $g(x_1) = ax_1 + b$. Since m_1 and m_2 are mutually nondominating it follows that a is negative. Define by g_2 the straight line which is parallel to g and which goes through z, i.e., $g_2(x_1) = ax_1 + b_2$. Since by assumption $D(d, m_1, m_2) \leq D(z, m_1, m_2)$ it follows that $f_2(d) \geq g_2(f_1(d))$. Since the slope a of g_2 is negative it follows that either $f_1(d) \geq f_1(z)$ or $f_2(d) \geq f_2(z)$ which is a contradiction to $d \prec z$, and thus, it must the that $D(d, m_1, m_2) > D(z, m_1, m_2)$.

Algorithm 2. $\{m_1, m_2, K\} := ArchiveUpdateMaxBulge2\ (p, K^0, m_1^0, m_2^0, \Delta)$

```
1: if $f_1(p) < f_1(m_1^0)$ then
2:     $m_1 := p$
3: else
4:     $m_1 := m_1^0$
5: end if
6: if $f_2(p) < f_1(m_2^0)$ then
7:     $m_2 := p$
8: else
9:     $m_2 := m_2^0$
10: end if
11: $\tilde{K} := K^0 \cup \{p\}$
12: $\tilde{M} := \max_{k\in\tilde{K}} D(k, m_1, m_2)$
13: $K := nondom(\{k \in \tilde{K} : D(k, m_1, m_2) \geq \tilde{M} - \Delta\})$
```

Theorem 2. *Using the definitions above, let $M > 0$, $\Delta \in \mathbb{R}_+$ with $M - \Delta > 0$, and let*

$$\lim_{i\to\infty} K_{\Delta_i} \to K_\Delta \tag{8}$$

for every sequence $(\Delta_i)_{i\in\mathbb{N}}$ with $\Delta_i < \Delta$ and $\Delta_i \to \Delta$ for $i \to \infty$. Then, if Algorithm 2 is used to update the sequences $K_l, m_1^{(l)}, m_2^{(l)}, l \in \mathbb{N}$, and under the assumptions made in Thm. 1 it holds with probability one

(a)

$$m_1^{(l)} \to p_1^* \in \arg\min_{x\in P_Q} f_1(x) \quad \textit{for } l \to \infty$$

(b)

$$d_H(F(K_\Delta), F(K_l)) \to 0 \quad \textit{for } l \to \infty$$

Proof. (a) Analogue to proof of Thm 1 (a).

(b) First we show that $dist(F(K_\Delta), F(K_l)) \to 0$ for $l \to \infty$ with probability one. Since K_l, $l \in \mathbb{N}$, is finite and K_Δ is compact it follows that

$$dist(F(K_\Delta), F(K_l)) = \max_{p\in K_\Delta} \min_{k\in K_l} \|F(p) - F(k)\|. \tag{9}$$

By (8) it is sufficient to consider points $p \in P_Q$ with $D(p, p_1^*, p_2^*) > M - \Delta$. Let p be such a point. By Thm. 1 it follows that $\tilde{M}_l$ (see line 12 of Alg. 2) converges to M with probability one. Further, since D and F are continuous it follows that there exists with probability one a neigborhood U of p and an integer l_0 such that

$$D(u, m_1^{(l)}, m_2^{(l)}) > \tilde{M}_l - \Delta, \quad \forall u \in U,\ \forall l \geq l_0. \tag{10}$$

By (4) it follows that there exists with probability one for every $j \in \mathbb{N}$ a point $p_{l_j} \in U \cap B_{1/j}(p) \cap Q$. By construction of Alg. 2 the point p_{l_j} will either be added to the archive (in that case denote $d_j := p_{l_j}$), or there already exists a point $d_j \in K_{l_j}$ which dominates p_{l_j}. Due to (10) the point d_j will only be discarded from the archive if in turn a dominated solution is found. By this and since $p_j \to p$ and thus $F(d_j) \to F(p)$ for $j \to \infty$ it follows that

$$dist(F(p), F(K_l)) = \min_{k \in K_l} \|F(p) - F(k)\| \to 0 \quad \text{with probability one,} \tag{11}$$

and the claim follows. It remains to show that also

$$dist(F(K_l), F(K_\Delta)) = \max_{k \in K_l} \min_{p \in K_\Delta} \|F(k) - F(p)\| \tag{12}$$

vanishes for $l \to \infty$ and in the probabilistic sense. For this we have to show that every point $x \in Q \backslash K_\Delta$ will be discarded (if added before) from the archive after finitely many steps, and that this point will never be added further on, both with probability one. Let $x \in Q \backslash K_\Delta$, that is, we have either (a) $D(x, p_1^*, p_2^*) < M - \Delta$ or (b) $x \notin P_Q$. First we consider case (a). Since the sequence $\tilde{M}_l \to M$ (see above) and by continuity of D there exists with probability one an integer l_0 with

$$D(x, m_1^{(l)}, m_2^{(l)}) < \tilde{M}_l - \Delta, \quad \forall l \geq l_0, \tag{13}$$

and by this, that x is not a member of K_l for $l \geq l_0$.

Next, let $x \notin P_Q$. By case (a) we can assume that $D(x, p_1^*, p_2^*) \geq M - \Delta > 0$. Since x is not a weak Pareto point there exists a point $p \in P_Q$ with $F(p) <_p F(x)$. By continuity of D and F, by part (a) of this theorem, and by Lemma 1 it follows that there exists a neighborhood U of p and an integer l_0 such that:

$$\begin{aligned} F(u) &<_p F(x), \quad \forall u \in U, \text{ and} \\ D(u, m_1^{(l)}, m_2^{(l)}) &> D(x, m_1^{(l)}, m_2^{(l)}), \quad \forall u \in U,\ \forall l \geq l_0 \\ D(u, m_1^{(l)}, m_2^{(l)}) &> \tilde{M} - \Delta, \quad \forall u \in U,\ \forall l \geq l_0. \end{aligned} \tag{14}$$

By (4) it follows that there exists with probability one an integer $j_0 > l_0$ such that that the candidate solution p_{j_0} is in $U \cap Q$. Further, by (14) and by construction of Alg. 2 it follows that p_{j_0} will be either added to the archive or that there already exists a point d which dominates p_{j_0}. In further iterations of the algorithm, this point is only discarded if a dominated solution is found (using (14) and Lemma 1). Since $\prec$ is transitive all these points dominate x, and hence is not a member of K_l for all integers $l \geq j_0$ with probability one, and the proof is complete.

Since the archiver in Alg. 2 accepts all points in K_Δ and does not discard them further on it follows that in the course of the computation $|K_l| \to \infty$ for $l \to \infty$. In order to prevent this, one could select a subset of K_l in each step, e.g., by the techniques proposed in [8] or other pruning techniques.

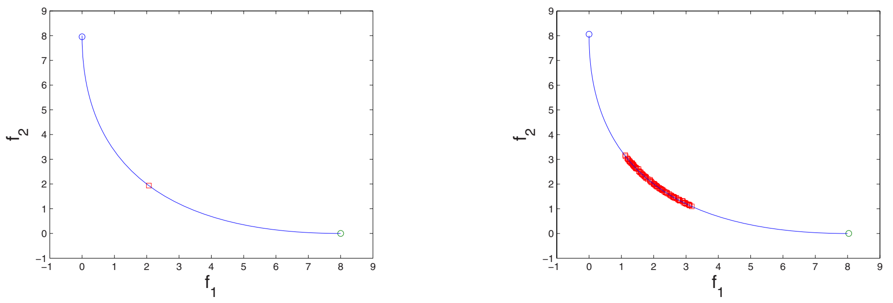


Fig. 2. Numerical results for MOP (15) with $N = 10,000$ randomly chosen points within $Q = [-2,2]^2$ for Alg. 1 (left) and for Alg. 2 for $\Delta = 0.2$. The circles represent the final extreme points, and the square(s) the approximation of the knee (region).

5 Numerical Results

Here we present some numerical results on two MOPs: a convex problem and an MOP ([14]) which has two optimal points with maximal bulge:

$$\begin{aligned} F_1 &: [-2,2]^2 \to \mathbb{R}^2 \\ F_1(x) &= \big((x_1-1)^2 + (x_2-1)^2, (x_1+1)^2 + (x_2+1)^2\big) \end{aligned} \tag{15}$$

and

$$\begin{aligned} F_2 &= (f_1, f_2) : [-1.5, 1.5]^2 \to \mathbb{R}^2 \\ f_1(x,y) &= \frac{1}{2}(\sqrt{1+(x+y)^2} + \sqrt{1+(x-y)^2} + x - y) + \lambda \cdot e^{-(x-y)^2} \\ f_2(x,y) &= \frac{1}{2}(\sqrt{1+(x+y)^2} + \sqrt{1+(x-y)^2} - x + y) + \lambda \cdot e^{-(x-y)^2} \end{aligned} \tag{16}$$

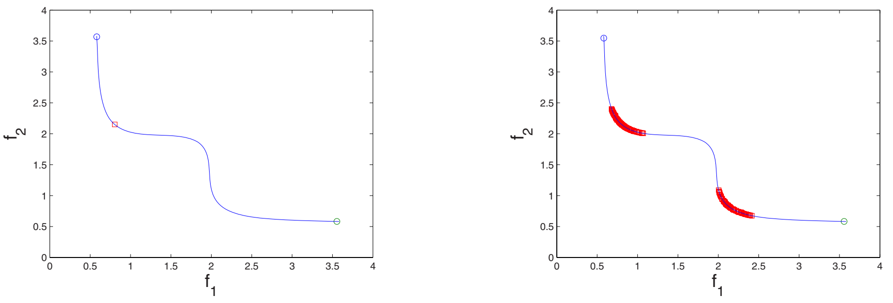


Fig. 3. Two numerical results for MOP (16) with $N = 10,000$ randomly chosen points within $Q = [-1.5, 1.5]^2$ for Alg. 1 (left) and for Alg. 2 (right) for $\Delta = 0.1$. The circles represent the final extreme points, and the square(s) the approximation of the knee (region).

For the generation of the sequence $(p_l)_{l\in\mathbb{N}}$ of candidate solutions we have taken a random search operator. Figures 2 and 3 show two numerical results—i.e., one result for every archiving strategy—for each of the models. MOP (16) contains two maximal bulges, and hence, the archiver $ArchiveUpdateMaxBulge1$ can only reach one of them (Fig. 3 (a)). However, this does not occur when using the second archiver (Fig. 3 (b)).

6 Conclusions and Future Work

In this paper we have proposed and investigated two update strategies for the approximation of knees respectively knee-regions of multi-objective optimization problems with stochastic search algorithms. The advantage of these methods is that they can be used either as standalone-algorithms together with any stochastic search procedure or integrated into any other archiving strategy (e.g., distance based ones) without causing additional function calls. We have demonstrated on two examples where we have used a random search operator that the novel strategies are capable of approximating the desired regions with reasonable effort.

For future research, there are mainly two points which have to be addressed. First, a generalization of the obtained results for $k > 2$ would be desirable. Further, the integration of the archivers into stochastic search procedures is of particular interest: since the archivers focus on a real subset of the Pareto front, a natural demand on the resulting algorithm is that it should be more efficient in terms of function calls than algorithms which aim for the approximation of the entire Pareto front. This is, however, ad hoc not straightforward since it is well-known that the approximation of the nadir points can be a challenging task itself.

Acknowledgements. The third author gratefully acknowledges support from the CONACyT project no. 45683-Y.

References

1. Branke, J., Deb, K., Dierolf, H., Osswald, M.: Finding knees in multi-objective optimization. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN VIII 2004. LNCS, vol. 3242, pp. 722–731. Springer, Heidelberg (2004)
2. Branke, J., Kaussler, T., Schmeck, H.: Guidance in evolutionary multi-objective optimization. Advances in Engineering Software 32, 499–507 (2001)
3. Das, I.: On characterizing the "knee" of the Pareto curve based on Normal Boundary Intersection. Structural Optimization 18, 107–115 (1999)
4. Deb, K.: Multi-objective evolutionary algorithms: introducing bias among pareto-optimal solutions, pp. 263–292 (2003)

5. di Pierro, F., Khu, S.F., Savic, D.A.: An investigation on preference order ranking scheme for multiobjective evolutionary optimization. IEEE Transactions on Evolutionary Computation 11(1), 17–45 (2007)
6. Handl, J., Knowles, J.: Exploiting the trade-off: the benefits of multiple objectives in data clustering. In: Proceedings of the Third International Conference on Evolutionary Multicriterion Optimization, pp. 547–560
7. Ishibuchi, H., Nojima, Y., Narukawa, K., Doi, T.: Incorporation of decision maker's preference into evolutionary multiobjective optimization algorithms. In: GECCO, pp. 741–742 (2006)
8. Kukkonen, S., Deb, K.: A fast and effective method for pruning of non-dominated solutions in many-objective problems. In: PPSN, pp. 553–562 (2006)
9. Mattson, C.A., Mullur, A.A., Messac, A.: Smart Pareto filter: Obtaining a minimal representation of multiobjective design space. Engineering Optimization 36, 721–740 (2004)
10. Mehnen, J., Trautmann, H.: Integration of expert's preferences in pareto optimization by desirability function techniques. In: Proceedings of the 5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering (CIRP ICME 2006) (2006)
11. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Dordrecht (1999)
12. Rachmawati, L., Srinivasan, D.: A multi-objective evolutionary algorithm with weighted-sum niching for convergence on knee regions. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 749–750 (2006)
13. Rachmawati, L., Srinivasan, D.: A Multi-Objective Genetic Algorithm with Controllable Convergence on Knee Regions. In: IEEE Congress on Evolutionary Computation, CEC 2006, pp. 1916–1923 (2006)
14. Witting, K., Hessel-von Molo, M.: Private communication (2006)

Appendix

Given points $p, a_1, a_N \in Q$, the distance of $F(p)$ to the straight line $\mathcal{L}(F(a_1), F(a_N))$ can be computed as follows: since $\mathcal{L}(F(a_1), F(a_N)) \subset \mathbb{R}^2$, it can be written as a function $g_1 : \mathbb{R} \to \mathbb{R}$, $g_1(x) = m_1 x + b_1$ with $m_1 = -(f_2(a_1) - f_2(a_N))/(f_1(a_N) - f_1(a_1))$, and $b_1 = f_2(a_1) - m_1 f_1(a_1)$, where the interpolation conditions $g_1(f_1(a_j)) = f_2(a_j), j \in \{1, N\}$, are used. To compute the distance of $F(p)$ and g_1 we define the auxiliary function $g_2(x) = m_2 x + b_2$ with $g_2(f_1(p)) = f_2(p)$ and which is orthogonal to g_1. Doing so, this leads to the coefficients $m_2 = -1/m_1$ and $b_2 = f_2(p) - m_2 f_1(p)$. The intersection of g_1 and g_2 is given by the point $S_p = (x_s, y_s)$ with $x_s = (b_2 - b_1)/(m_1 - m_2)$, $y_s = m_2 x_s + b_2$, and thus, we have

$$dist(F(p), \mathcal{L}(F(a_1), F(a_N))) = \|F(p) - S_p\| \tag{17}$$

Approximate Solutions in Space Mission Design

Oliver Schütze[1], Massimiliano Vasile[2], and Carlos A. Coello Coello[1]

[1] CINVESTAV-IPN, Computer Science Department, Mexico City, Mexico
{schuetze,ccoello}@cs.cinvestav.mx
[2] University of Glasgow, Department of Aerospace Engineering, Glasgow, Scotland
m.vasile@aero.gla.ac.uk

Abstract. In this paper, we address multi-objective space mission design problems. We argue that it makes sense from the practical point of view to consider in addition to the 'optimal' trajectories (in the Pareto sense) also approximate or nearly optimal solutions since this can lead to a significant larger variety for the decision maker. For this, we suggest a novel MOEA which is a modification of the well-known NSGA-II algorithm equipped with a recently proposed archiving strategy which aims for the storage of the set of approximate solution of a given MOP. Using this algorithm we will examine several space missions and demonstrate the benefit of the novel approach.

1 Introduction

In a variety of applications in industry and finance a problem arises that several objective functions have to be optimized concurrently leading to *multi-objective optimization problems* (MOPs). For instance, in space mission design, which we address here, there are two crucial aims for the realization of a transfer: minimization of flight time and fuel consumption of the spacecraft ([2], [13], [11], [10]). The scope of this paper is (a) to show that it makes sense to consider in addition to the 'optimal' trajectories also approximate solutions since by this the decision maker (DM) is offered a much larger variety of possibilities, and (b) to present one way to compute this enlarged set of interest with reasonable effort. As a motivating example for (a) we consider the MOP in Section 4.2 which is a model for the transfer from Earth to Mercury, and the following two points x_i with images $F(x_i), i = 1, 2$:

$$\begin{aligned} x_1 &= (782,\ 1288,\ 1788)\ , & F(x_1) &= (0.462,\ 1001.7) \\ x_2 &= (1222,\ 1642,\ 2224), & F(x_2) &= (0.463,\ 1005{,}3) \end{aligned}$$

The two objectives are the propellant mass fraction—i.e., the portion of the vehicle's mass which does not reach the destination—and the flight time (in days). In the domain, the first parameter is of particular interest: it determines the departure time from the Earth (in days after 01.01.2000). $F(x_1)$ is less than $F(x_2)$ in both components, and thus, x_1 can be considered to be 'better' than x_2. However, note that the difference in image space is small: the mass fraction of the two solutions differs by 0.001 which makes 0.1% of the total mass, and the flight time differs by four days for a transfer which takes almost three years.

In case the DM is willing to accept this deterioration, it will offer him/her a second choice in addition to x_1 for the realization of the transfer: while the two solutions offer 'similar' characteristics in image space this is not the case in the design space since the starting times for the two transfers differ by 440 days.

The identification of the two solutions would be a fundamental requirement during the preliminary design of a space mission. In fact, in order to increase the reliability of the design, the mission analysts would need to identify one or more back-up solutions, possibly with identical cost, for each baseline solution. Furthermore, for each mission opportunity (i.e., each launch date) rather than an optimal solution, it is generally required to identify a set of nearly optimal ones, possibly all with similar cost. Such a set would represent a so called *launch window*, since for each solution in the set a launch would be possible. Designing for the suboptimal points further increases the reliability of the mission since it gives the freedom to deviate from the chosen design point with little or no penalty. This holds true also for Pareto optimal solutions. It is therefore desirable to have a whole range of nearly Pareto optimal solutions for each Pareto point.

The field of evolutionary multi-objective optimization is well-studied and MOEAs have been successfully applied in a number of domains, most notably engineering applications ([1]). Approximate solutions in multi-objective optimization have been studied by many researchers so far (e.g., [7], [14], [6]). A first attempt to investigate the benefit of considering approximate solutions in space mission design has been done in ([12]), albeit for the single-objective case.

The additional consideration of (all) approximate solutions in multi-objective space mission design problems is new and will be addressed in this paper. Crucial for this approach is the efficient computation of the enlarged set of 'optimal' points since in many cases the 'classical' multi-objective approach is a challenge itself. For this, we will propose an algorithm which is based on the well-known NSGA-II ([3]) but equipped with an archiving strategy which was designed for the current purpose. Note that 'classical' archiving/selection strategies—e.g., the ones in [4], [9], [6], [5], or the one NSGA-II uses—store sets of mutually non-dominating points (which means that e.g. the points x_1 and x_2 in the above example will never be stored *jointly*). That is, these selection mechanisms—though they accomplish an excellent job in approximating the efficient set—can not be taken for our purpose.

The remainder of this paper is organized as follows: in Section 2, we give the required background which includes the statement of the space mission design problem under consideration. In Section 3, we propose a new genetic algorithm for the computation of the set of approximate solutions and present further on in Section 4 some numerical results. Finally, we conclude in Section 5.

2 Background

2.1 Multi-Objective Optimization

In the following we consider continuous multi-objective optimization problems

$$\min_{x \in Q} \{F(x)\}, \qquad \text{(MOP)}$$

where $Q \subset \mathbb{R}^n$ is compact and F is defined as the vector of the objective functions $F : Q \to \mathbb{R}^k, \quad F(x) = (f_1(x), \ldots, f_k(x))$, with $f_i : Q \to \mathbb{R}$.

Definition 1. *Let $v, w \in Q$. Then the vector v is* less than w *($v <_p w$), if $v_i < w_i$ for all $i \in \{1, \ldots, k\}$. The relation $\leq_p$ is defined analogously. $y \in Q$ is* dominated *by a point $x \in Q$ ($x \prec y$) with respect to (MOP) if $F(x) \leq_p F(y)$ and $F(x) \neq F(y)$. $x \in Q$ is called a* Pareto optimal point *or* Pareto point *if there is no $y \in Q$ which dominates x.*

The set of all Pareto optimal solutions is called the *Pareto set* (denoted by P_Q). The image of the Pareto set is called the *Pareto front*. We now define another notion of dominance which we use to define approximate solutions and the set of interest:

Definition 2. *Let $\epsilon = (\epsilon_1, \ldots, \epsilon_k) \in \mathbb{R}^k_+$ and $x, y \in Q$. x is said to ϵ-dominate y ($x \prec_\epsilon y$) with respect to (MOP) if $F(x) - \epsilon \leq_p F(y)$ and $F(x) - \epsilon \neq F(y)$. x is said to $-\epsilon$-dominate y ($x \prec_{-\epsilon} y$) with respect to (MOP) if $F(x) + \epsilon \leq_p F(y)$ and $F(x) + \epsilon \neq F(y)$.*

Definition 3. *Denote by $P_{Q,\epsilon}$ the set of points in $Q \subset \mathbb{R}^n$ which are not $-\epsilon$-dominated by any other point in Q, i.e., $P_{Q,\epsilon} := \{x \in Q | \nexists y \in Q : y \prec_{-\epsilon} x\}$.*

The set $P_{Q,\epsilon}$ contains all ϵ-efficient solutions, i.e., solutions which are optimal up to a given (small) value of ϵ. Fig. 1 gives two examples.

Alg. 1 gives an archiving strategy which aims for the approximation of $P_{Q,\epsilon}$, where A_0 is a given archive, p a candidate solution, $\Delta \in \mathbb{R}^k_+$ the discretization parameter, and $B(y, \Delta) := \{x \in \mathbb{R}^k : |x_i - y_i| \leq \Delta_i, i = 1, .., k\}$. See [10] for the related discussion.

2.2 The Design Problem

The examples we analyze are taken from two classes of typical problems in space trajectory design: a bi-impulsive transfer from the Earth to the asteroid Apophis, and a low-thrust multi-gravity assist transfer.

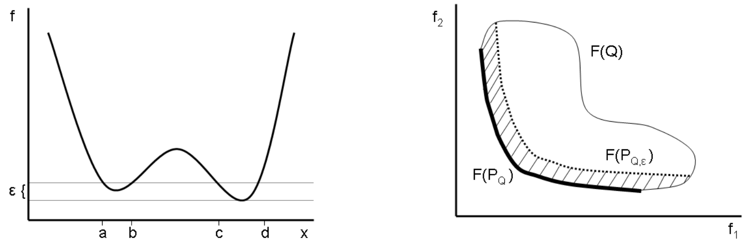


Fig. 1. Two different examples for sets $P_{Q,\epsilon}$. Left for $k = 1$ and in parameter space with $P_{Q,\epsilon} = [a, b] \cup [c, d]$. Right an example for $k = 2$ in image space.

Algorithm 1. $A := ArchiveUpdateP_{Q,\epsilon}\ (p, A_0, \Delta)$

```
Require: population $P$, archive $A_0$, $\Delta \in \mathbb{R}_+$, $\Delta^* \in (0, \Delta)$
Ensure: updated archive $A$
1: $A := A_0$
2: if $\nexists a_1 \in A:\ a_2 \prec_{-\epsilon} p$ and $\nexists a_2 \in A:\ F(p) \in B(F(a_2), \Delta^*)$ then
3:     $A := A \cup \{p\}$
4:     for all $a \in A$ do
5:         if $p \prec_{-(\epsilon+\Delta)} a$ then
6:             $A := A \backslash \{a\}$
7:         end if
8:     end for
9: end if
```

Bi-impulse Problem. For the bi-impulsive case, the propellant consumption is a function of the velocity change, or Δv, required to depart from the Earth and to rendezvous with a given celestial body. Both the Earth and the target celestial body are point masses with the only source of gravity attraction being the Sun. Therefore, the spacecraft is assumed to be initially at the Earth, flying along its orbit. The first velocity change, or Δv_1, is used to leave the orbit of the Earth and put the spacecraft into a transfer orbit to the target. The second change in velocity, or Δv_2, is then used to inject the spacecraft into target's orbit. The two Δv's are a function of the positions of the Earth and the target celestial body at the time of departure t_0 and at the time of arrival $t_f = t_0 + T$, where T is the time of flight. Thus, the MOP under consideration has two objective functions $f_1(x) = \Delta v_1 + \Delta v_2$ and $f_2(x) = T$, with the solution vector $x = [t_0, T]^T$.

MLTGA Problem. It is here proposed to use a particular model for multiple gravity assist low-thrust trajectories (MLTGA). Low-thrust arcs are modeled through a shaping approach based on the exponential sinusoid proposed in [8]. The spacecraft is assumed to be moving in a plane subject to the gravity attraction of the Sun and to the control acceleration of a low-thrust propulsion engine[13]. Gravity manoeuvres are modeled through a powered swing-bys approximation[13]: a pair of low-thrust arcs are linked through a Δv manoeuvre when the gravity of the swing-by planet is not strong enough to gain the required change in velocity. As for the bi-impulsive case, we are interested in the minimization of two objectives: the propellant mass fraction and the flight time. The first objective is $f_1(x) = 1 - e^{-(\frac{\Delta V_{GA}+\Delta V_0}{g_0 I_{sp1}} + \frac{\Delta V_{LT}}{g_0 I_{sp2}})}$ with the solution vector[11] $x = [t_0, T_1, k_{2,1}, n_1, ..., T_i, k_{2,i}, n_i, ..., T_N, k_{2,N}, n_N]^T$. Where ΔV_{GA} is the sum of all the $\Delta V s$ (variation in velocity) required to correct every gravity assist manoeuvre, ΔV_0 is the departure manoeuvre, while ΔV_{LT} is the sum of the total ΔV of each low-thrust leg. Then, $k_{2,i}$ is the $i-th$ shaping parameter for the exponential sinusoid and n_i the number of revolutions around the Sun, t_0 is the departure time and T_i the transfer time from planet i to planet $i+1$. The two specific impulses I_{sp1} and I_{sp2} are respectively for a chemical engine and for a low-thrust engine and g_0 is the gravity acceleration on the surface of the Earth. For the tests in this paper, we used $I_{sp1} = 315$s and $I_{sp2} = 2500$s. The second objective function is $f_2(x) = t_N - t_0$ with t_N the time of arrival at destination.

3 A Genetic Algorithm for the Computation of $P_{Q,\epsilon}$

In this section we propose a MOEA which aims for the computation of the set of approximate solutions, $P_{Q,\epsilon}$-NSGA-II, which is a hybrid of NSGA-II ([3]) and the archiver $ArchiveUpdateP_{Q,\epsilon}$. Further, in order to be able to compare the obtained solutions with another strategy, we introduce a performance metric.

The Algorithm. The algorithm we propose in the following is based on NSGA-II. We have decided to take this one as our baseline algorithm for two reasons. First, this algorithm is well-known and has been found to be very efficient. Second, we think that the elements which constitute NSGA-II fit nicely to our context: a (finite) archive A containing points which are mutually non-$(-\epsilon)$-dominating can be viewed as a set of Pareto fronts with different ranks, and also in the current setting the first front (i.e., the non-dominated front) should be given the priority since (i) improvement of the current set is clearly an objective and—in case the solutions are already near to P_Q—a local search around P_Q (e.g., mutation) is a search within $P_{Q,\epsilon}$. Thus, we have decided to adopt the ranking from NSGA-II, as well as the crowding distance in order to maintain diversity, and the genetic operators since they are proven to be well-suited for continuous problems.

The algorithm $P_{Q,\epsilon}$-NSGA-II reads as follows: the initial offspring $\mathcal{O} \subset Q$ is chosen at random, and the first archiver is set to $\mathcal{A}_0 := ArchiveUpdateP_{Q,\epsilon}(\emptyset, \mathcal{O}_0, \Delta)$. Alg. 2 describes how to obtain the subsequent archives $\mathcal{A}_{l+1}$ from $\mathcal{A}_l$. Hereby the function $Select()$ picks $n_p/2$ elements from $\mathcal{A}$ at random, if $|\mathcal{A}| \leq n_p/2$ then $\mathcal{C} := \mathcal{A}$ is chosen (n_p denotes the population size). The next three operators are as in NSGA-II: $DominationSort()$ assigns rank and crowding distance to $\mathcal{C}$, $TournamentSelection()$ performs the tournament selection, and $GeneticOperator()$ performs simulated binary crossover and polynomial mutation on $\mathcal{P}$. Finally, the archive $\mathcal{A}_l$ is updated by $\mathcal{O}$ using $ArchiveUpdateP_{Q,\epsilon}$ leading to the new archive $\mathcal{A}_{l+1}$.

The new algorithm is in fact very close to NSGA-II, merely the selection strategy to keep the 'promising' points of the search has changed (by adding an archive to NSGA-II). Recall that the motivation for the storage of approximate solutions is to obtain in addition to the 'optimal' points also points which are close to these points in image space but which differ significantly in parameter space. Thus, it is desired to maintain a certain diversity in parameter space, and that is why the chromosomes $\mathcal{C}$ are chosen randomly from the current archive by $Select()$.

Performance Metric. In order to be able to compare the results of different algorithms, or just two sets A and B, we propose to use the following metric:

$$\mathcal{C}_{-\epsilon}(A, B) := |\{b \in B \ : \ \exists a \in A : a \prec_{-\epsilon} b\}|/|B|, \tag{1}$$

Algorithm 2. Iteration step of $P_{Q,\epsilon}$-NSGA-II

```
Require: archive $\mathcal{A}_l$, $\Delta \in \mathbb{R}_+$, population size $n_p$
Ensure: updated archive $\mathcal{A}_{l+1}$
1: $\mathcal{C} := Select(\mathcal{A}_l, n_p/2)$
2: $\mathcal{C}' := DominationSort(\mathcal{C})$
3: $\mathcal{P} := TournamentSelection(\mathcal{C}')$
4: $\mathcal{O} := GeneticOperator(\mathcal{P})$
5: $\mathcal{A}_{l+1} := ArchiveUpdateP_{Q,\epsilon}(\mathcal{A}_l, \mathcal{O}, \Delta)$
```

which is a straightforward extension of the *set coverage metric* suggested in [15]. Analogue to the original metric, $\mathcal{C}_{-\epsilon}(A, B)$ is an unsymmetric operator which aims to get an idea of the relative coverage of the two solution sets.

4 Numerical Results

Here we present some numerical results coming from two different settings. For the internal parameters (e.g., mutation probability) of NSGA-II we have followed the suggestions made in [3], and have taken the same values for $P_{Q,\epsilon}$-NSGA-II.

4.1 Two Impulse Transfer to Asteroid Apophis

For the bi-impulse problem we analyze an apparently simple case: the direct transfer from the Earth to the asteroid Apophis. The contour lines of the sum of the two Δv's is represented in Fig.2 (a) for the parameters $t_0 \in [3675, 10500]^T$ MJD2000 and $T \in [50, 900]$ days. The intervals for t_0 and T were chosen in such a way that a wide range of launch opportunities are included. The solution space presents a large number of local minima. Many of them are nested, very close to each other and with similar values. For each local minimum, there can be a different front of locally Pareto optimal solutions. The best known approximation of the global Pareto front is represented in Fig. 2 (b) and was obtained with

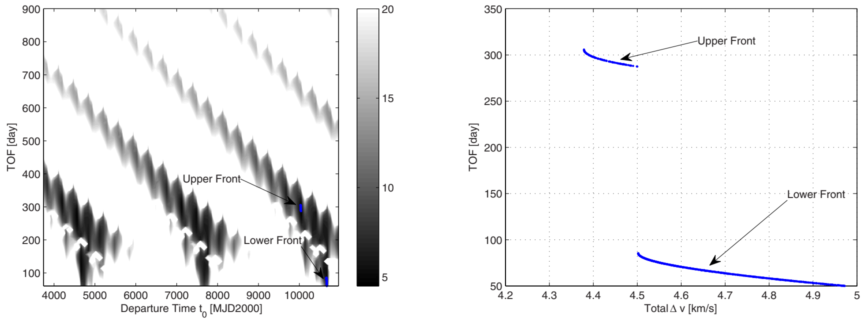
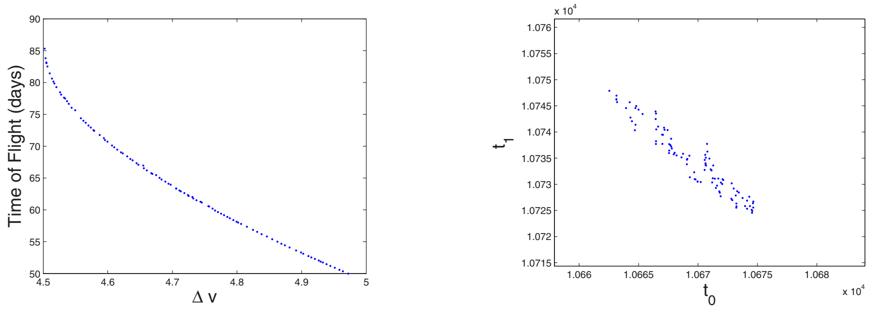


Fig. 2. a) Earth-Apophis search space, b) Pareto front



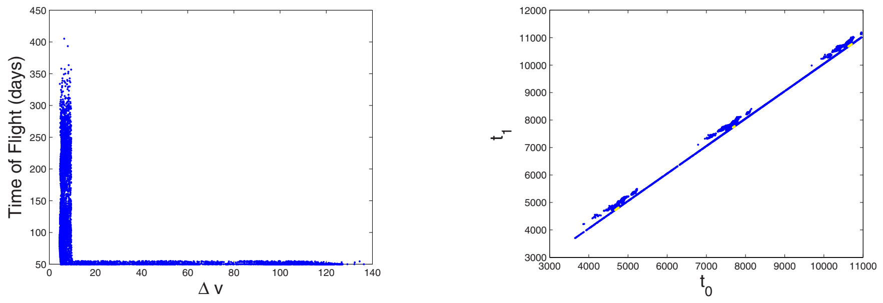
(a) Image space (b) Parameter space

Fig. 3. Numerical result for Example 2 using NSGA-II, $t_1 := t_0 + T$

an extension to MOPs of the algorithm described in [12]. It is a disjoint front corresponding to two basins of attraction of two minima, see Fig. 2 (a).

The two basins of attraction present similar values of the first objective function. Converging to the upper front is therefore quite a challenge since the lower front has a significantly lower value of the second objective function. It is only when the optimizer converges to a point in the vicinity of the local minimum of the upper front that the latter becomes not dominated by the lower front. The upper front contains the global minimum with a total $\Delta v = 4.3786$ k/s while the lower front contains only a local minimum. It should be noted that, though the front in Fig. 2b) is the global one, it represents only two launch opportunities. Furthermore for each launch opportunity we would need to characterize the space around each of the Pareto optimal point.

Figure 3 shows a result from NSGA-II, where the lower front has been found. When using $P_{Q,\epsilon}$-NSGA-II using the same parameter values as for NSGA-II and $\epsilon = (5, 5)$, which seems to be acceptable for this mission, a much broader variety of solutions is offered regardless of the upper front, as shown in Figure 4



(a) Image space (b) Parameter space

Fig. 4. Numerical result for Example 2 using $P_{Q,\epsilon}$-NSGA-II

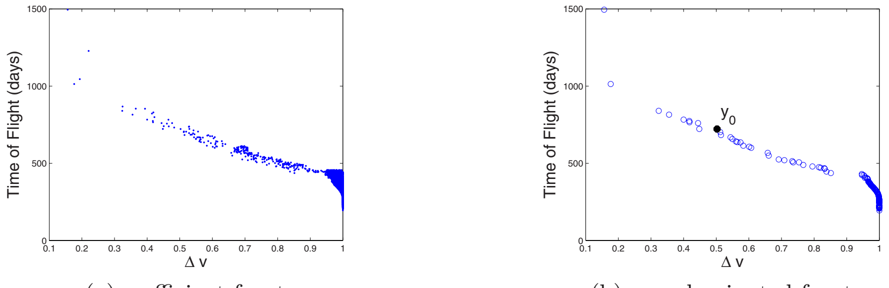
(note the difference of the scales). For instance, for the obtained solution c_0 with $F(c_0) = y_0 = (5, 50)$ there are three clusters of solutions which offer a similar cost and which are located around the points $c_1 = (t_0 = 4700, T = 50)$, $c_2 = (7700, 50)$, and $c_3 = (10700, 50)$. That is, the starting times of the transfer differ by a total of 6000 days. In contrast, the maximal difference according to t_0 of *all* the solution displayed in Figure 3 is given by 35 days.

Note that, compared to the accurate solution of the global Pareto front, the extended solution set offers, as required, not only more launch opportunities but also the whole neighboring solutions for each one of them.

4.2 Sequence EVMe

For the MLTGA problem we consider a relatively simple but significant case: the sequence Earth – Venus – Mercury (EVMe). For such a mission we have chosen to allow a deterioration of 5% of the mass fraction and of 20 days transfer time compared to an optimal trajectory which leads to $\epsilon = (0.05, 20)$. Figure 5 shows a numerical result of $P_{Q,\epsilon}$-NSGA-II for 100 generations with population size 100 (i.e., the size of $\mathcal{P}$ in Alg. 2) and $\Delta = \epsilon/3$, which took several minutes on a standard PC. To compare the result and since so far no such algorithm exists we have taken a random search procedure coupled with $ArchiveUpdateP_{Q,\epsilon}$. For $N_R = 10,000$ randomly chosen points we obtain (averaged of 20 test runs) $\mathcal{C}_{-\epsilon}(A_N, A_R) = 0.4739$ and $\mathcal{C}_{-\epsilon}(A_R, A_N) = 0$, where A_N denotes the result from $P_{Q,\epsilon}$-NSGA-II and A_R the result coming from the random search procedure. For $N_R = 100,000$ the result of the random search procedure can still not compete with the same MOEA result: $\mathcal{C}_{-\epsilon}(A_N, A_R) = 0.4261$, $\mathcal{C}_{-\epsilon}(A_R, A_N) = 0$.

Interesting for every non-dominated point x_0 with $F(x_0) = y_0$ of an archive A is the set $N(y_0, \epsilon, A) := \{a \in A \ : \ F(a) \in B(y_0, \epsilon)\}$, where $B(y, \epsilon) := \{x \in \mathbb{R}^k \ : \ |x_i - y_i| \leq \epsilon_i,\ i = 1, .., k\}$, i.e., the set of solutions in A those images are 'close' to y_0. Since in this design problem the starting date t_0 of the transfer is of particular interest one can e.g. distinguish the entries in $N(y_0, \epsilon, A)$ by the value of t_0. For instance, the final archive displayed in Figure 5 (a) consists of



(a) ϵ-efficient front (b) non-dominated front

Fig. 5. Numerical result for sequence EVMe. Left the final archive and right the set of non-dominated solutions.

3650 solutions whereof 106 are non-dominated. The maximal difference of the value of t_0 for a point y_0 inside $N(y_0, \epsilon, A)$ is 449 days, and for 23 solutions this maximal difference is larger than one year (including also values Δt_0 of several days or months which can be also highly interesting for the decision making process). Hence, the number of options for the DM is enlarged significantly in this example.

The consideration above leads to a natural way of presenting the large amount of data to the DM: it is sufficient to present the non-dominated front as in the 'classical' multi-objective case. When the DM selects one solution y_0 the set $N(y_0, \epsilon, A)$ can be displayed, ordered by the value of t_0 (see Figure 5 (b)).

5 Conclusion

We have considered two multi-objective space mission design problems and shown, that it is desirable to identify not only the Pareto set, but also a number of approximate solutions. In particular, it was shown that each part of the Pareto set belongs to a different launch window. In order to increase the reliability of the mission design, it is required to have a wide launch window (i.e., a large number of solutions with similar cost) and one or more back-up launch windows. In order to address this problem, we have proposed a new variant of an existing MOEA which aims for the computation of $P_{Q,\epsilon}$. As an example of its effectiveness, we have considered two design problems. The results indicate that the novel approach accomplishes its task within reasonable time and that the idea to include approximate solutions is indeed beneficial since in all cases the enlarged set of solutions offered a much larger variety to the DM. Despite these promising numerical results, however, more work is required for the design of a more efficient MOEA for the approximation of $P_{Q,\epsilon}$ which will be part of future work.

Acknowledgements. The third author gratefully acknowledges support from the CONACyT project no. 45683-Y.

References

1. Coello Coello, C.A., Lamont, G., Van Veldhuizen, D.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, Heidelberg (2007)
2. Coverstone-Caroll, V., Hartmann, J.W., Mason, W.M.: Optimal multi-objective low-thrust spacecraft trajectories. Computer Methods in Applied Mechanics and Engineering 186, 387–402 (2000)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
4. Hanne, T.: On the convergence of multiobjective evolutionary algorithms. European Journal Of Operational Research 117(3), 553–564 (1999)
5. Knowles, J., Corne, D.: Bounded Pareto Archiving: Theory and Practice. In: Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 39–64. Springer, Heidelberg (2004)

6. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary Computation 10(3), 263–282 (2002)
7. Loridan, P.: ϵ-solutions in vector minimization problems. Journal of Optimization, Theory and Application 42, 265–276 (1984)
8. Petropoulos, A.E., Longuski, J.M., Vinh, N.X.: Shape-based analytical representations of low-thrust trajectories for gravity-assist applications. In: AAS/AIAA Astrodynamics Specialists Conference, AAS Paper 99-337, Girdwood, Alaska (1999)
9. Rudolph, G., Agapie, A.: Convergence properties of some multi-objective evolutionary algorithms. In: Proceedings of the 2000 Conference on Evolutionary Computation, vol. 2, pp. 1010–1016 (2000)
10. Schütze, O., Coello Coello, C.A., Tantar, E., Talbi, E.-G.: Computing finite size representations of the set of approximate solutions of an MOP with stochastic search algorithms. In: The Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008) (to appear, 2008)
11. Schütze, O., Vasile, M., Junge, O., Dellnitz, M., Izzo, D.: Designing optimal low thrust gravity assist trajectories using space pruning and a multi-objective approach. In: Engineering Optimization (to appear, 2008)
12. Vasile, M., Locatelli, M.: A hybrid multiagent approach for global trajectory optimization. Journal of Global Optimization (to appear, 2008)
13. Vasile, M., Schütze, O., Junge, O., Radice, G., Dellnitz, M.: Spiral trajectories in global optimisation of interplanetary and orbital transfers. Ariadna study report ao4919 05/4106, contract number 19699/nl/he, European Space Agency (2006)
14. White, D.J.: Epsilon efficiency. Journal of Optimization Theory and Applications 49(2), 319–337 (1986)
15. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, ETH Zurich, Switzerland (1999)

A Local Search Based Evolutionary Multi-objective Optimization Approach for Fast and Accurate Convergence

Karthik Sindhya[1,3], Kalyanmoy Deb[1,3], and Kaisa Miettinen[2]

[1] Department of Mechanical Engineering
Indian Institute of Technology Kanpur, PIN 208016, India
ksindhya@iitk.ac.in, deb@iitk.ac.in

[2] Department of Mathematical Information Technology, P.O. Box 35 (Agora),
FI-40014 University of Jyväskylä, Finland
kaisa.miettinen@jyu.fi

[3] Department of Business Technology, Helsinki School of Economics, P.O. Box 1210,
FI-00101 Helsinki, Finland

Abstract. A local search method is often introduced in an evolutionary optimization technique to enhance its speed and accuracy of convergence to true optimal solutions. In multi-objective optimization problems, the implementation of a local search is a non-trivial task, as determining a goal for the local search in presence of multiple conflicting objectives becomes a difficult proposition. In this paper, we borrow a multiple criteria decision making concept of employing a reference point based approach of minimizing an achievement scalarizing function and include it as a search operator of an EMO algorithm. Simulation results with NSGA-II on a number of two to four-objective problems with and without the local search approach clearly show the importance of local search in aiding a computationally faster and more accurate convergence to Pareto-optimal solutions. The concept is now ready to be coupled with a faster and more accurate diversity-preserving procedure to make the overall procedure a competitive algorithm for multi-objective optimization.

1 Introduction

Evolutionary multi-objective optimization (EMO) algorithms are often criticized for their lack of a theoretical convergence proof to the true Pareto-optimal front. Although theoretical time complexity estimates of certain specific EMO algorithms exist [15] in solving specific test problems, in most problems a proof of convergence with a finite computational effort is missing. However, EMO algorithms also lack a theoretical proof of convergence to even on a local Pareto-optimal front. Moreover, a past study [14] has demonstrated and argued that EMO algorithm with a finite size archive for storing non-dominated solutions may allow an evolving population to fluctuate (convergence to the Pareto-optimal front followed by a departure of some solutions out of the front). This phenomenon can happen due to constant emphasis of diversity maintenance pursued in EMO. To make the diversity among obtained non-dominated solutions better, a Pareto-optimal solution may be sacrificed to accept a non-Pareto-optimal solution.

Despite the lack of theoretical convergence properties of EMO algorithms, they are increasingly being used in many applications, due to reasons such as, user's satisfaction with a near Pareto-optimal solution, difficulty in implementation of a local search procedure in a multi-objective context and in discrete and combinatorial optimization problems, optimality of a solution is impossible to verify. There exist many other practical optimization problems for which solutions close to the true Pareto-optimal front are desired with as low a computational effort as possible.

The use of local search in EMO has enjoyed a lot of attention in recent past, to make EMO algorithms converge faster on to the true Pareto-optimal front [3]. Here, we briefly mention some representative studies. A hybrid algorithm (S-MOGLS) using weighted sum of multiple objectives as fitness function was proposed in [9]. A neighborhood search (NS) was used as a local search which was then applied to all offspring solutions generated by NSGA-II. The algorithm C-MOGLS developed in [17] combined cellular multi-objective genetic algorithm and NS as a local search. The local search in this approach was applied using weighted sum of multiple objectives as fitness function to all non-dominated solutions in each generation. A new genetic local search algorithm which uses hybridization of recombination operators with a neighborhood based local search was presented in [10]. A random utility function was optimized locally in this algorithm. The local search was applied on all offspring solutions. In [7], a weighted sum of multiple objectives as fitness function was used and two approaches were presented hybridizing NSGA-II, a posteriori approach in which the local search is applied on all non-dominated solutions obtained after the NSGA-II simulation and an online approach in which the local search is applied to all offspring generated in each generation of NSGA-II. M-PAES which is a population version of multi-objective evolution strategy (PAES) was proposed in [11]. The search is enhanced by the use of (1+1)-ES as a local search. Furthermore, a new hybrid algorithm which uses Pareto descent method (PDM) as local search method was proposed in [8]. PDM finds feasible Pareto descent directions by solving computationally inexpensive linear programming problems.

Based on these studies and others from the literature, we observe two main approaches of using a local search with EMO. First, most studies implement local search as a refinement of solutions found by EMO. Second, when implemented within EMO, a local search is usually applied to all offspring solutions with a naive neighborhood based procedure. Not much effort has been given in borrowing more effective multiple criteria decision making (MCDM) [1] ideas in local search. In this paper, we propose the idea of one such hybrid approach in which EMO and local search are coupled and the latter is used to solve an augmented achievement scalarizing function (ASF). Since the solution to an augmented ASF is always a properly Pareto-optimal solution, the overall hybrid approach is shown to have a better convergence property than the EMO procedure alone.

The rest of this paper is organized as follows. In Section 2, we discuss the motivation of using a local search as a part of multi-objective optimization. The proposed hybrid approach is described in Section 3. Then, Section 4 describes results obtained by the hybrid approach and compared with original NSGA-II on test problems and briefly discusses possible approaches to ensure diversity in a hybrid approach. Finally, conclusions are drawn in Section 5.

2 Local Search in EMO

A local search is usually applied to improve the solution(s) obtained by an approximate optimizer. Here, we discuss the motivation for using a local search in a complex optimization task.

Optimization in practice involves objective function landscapes which may be multimodal, nondifferentiable, discrete, and involve many other complexities. It is unrealistic to expect a single optimization algorithm to be computationally efficient in handling different vagaries of function landscapes. A combination of two types of algorithms – an approximate global optimizer and an accurate local search – is one way to tackle the problem. The global optimizer searches the entire landscape to find the most promising region(s) with multiple points, while the local search begins its search from a particular solution and converges to a locally optimal solution. Thus, the roles of global and local optimizers are used to negotiate different function landscapes. Both optimization tasks are important and a balance of the extent of their searches is necessary for the overall procedure to converge to the true globally optimal solution.

In this study, we consider a population-based evolutionary algorithm as a global optimizer and a (gradient-based) mathematical programming method as a local search procedure, as they fit well with the above description of local and global optimizers. There are at least two different ways they can be hybridized: a *serial* and a *concurrent* approach. In a serial approach, global and local searches are applied serially one after the other with appropriate termination conditions. This switchover to local search from the global optimizer is not easy to fix a priori on any unknown problem, as a delay in terminating global solver can consume excess function evaluations and an early termination shall yield a locally optimal solution. For terminating a local search, a standard procedure such as the error in violations of Karush-Kuhn-Tucker (KKT) [12] conditions of optimality to be within a limit can be used. In the concurrent approach, local search is embedded within a global optimizer so that some or all intermediate solutions are modified by the local search. For example, in the EMO framework, the local search may be considered as an additional EMO operator which attempts to bring an intermediate solution to a locally Pareto-optimal solution. The termination criterion of the local search can be a standard one (as described above), but the termination of the global optimizer need not be ad-hoc, as in the serial approach, but can be based on whether there is an improvement in the locally optimal solutions over past few iterations. Due to the above advantages, we restrict ourselves to the concurrent approach of hybridization in this study.

In the case of solving single-objective optimization problems, the objective function to be optimized in the local search can be the same as that used in the global optimizer, as the main goal in this task is to find the global optimum of a single function landscape. Figure 1 illustrates this aspect. However, in handling multiple conflicting objectives, a local search faces an additional difficulty of choosing an appropriate single objective for its search. Since multiple conflicting objectives are of interest here, it is not fair to choose one particular objective function among the conflicting ones for the local search.

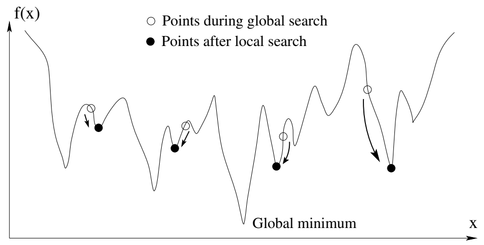


Fig. 1. Local and global searches can use the same objective function in a single-objective optimization

Thus, we realize that an implementation of a local search is nontrivial yet important in the context of multi-objective optimization. Despite, the existence of many local search approaches described in Section 1, a directed and computationally faster optimization approach is necessary for the local search.

In the following section, we suggest a widely used reference point based approach [18] from the MCDM field to be used in local search.

3 Proposed Local Search Based EMO

We propose a hybrid approach where we use the NSGA-II method [5] as the EMO procedure and hybridize it with an ASF (based on a reference point) which is solved with a local search method.

In the t-th generation of the NSGA-II procedure, an offspring population Q_t is created by using selection, recombination and mutation operators from the parent population P_t. Thereafter, each member of Q_t is evaluated and checked with a probability p_l for its improvement with the local search procedure. After the local search operations are performed, parent and offspring populations are combined together and a non-dominated sorting is performed. Thereafter, the NSGA-II procedure continues as usual.

The local search is started from an offspring solution $\mathbf{y}$ (having objective vector $\mathbf{f}(\mathbf{y})$). The local search procedure minimizes the following augmented achievement scalarizing function [18]:

$$\begin{aligned}&\text{minimize}_{\mathbf{x}} \max_{i=1}^{M} \frac{f_i(\mathbf{x})-z_i}{f_i^{\max}-f_i^{\min}} + \rho \sum_{j=1}^{M} \frac{f_j(\mathbf{x})-z_j}{f_j^{\max}-f_j^{\min}},\\ &\text{subject to } \mathbf{x} \in \mathcal{S},\end{aligned} \quad (1)$$

where $\mathcal{S}$ is the feasible decision variable space, $\mathbf{z}$=$\mathbf{f}(\mathbf{y})$ is the so-called reference point, and $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum objective values of the parent population P_t used for normalization.

By solving (1) we project the reference point (i.e. the solution produced by the EMO algorithm) onto the Pareto optimal front. The second term in the objective function (1) ensures that the local search will converge to a properly Pareto-optimal solution [16,18]. A small value of $\rho = 10^{-2}$ is used in this study. This local search procedure allows a directed search dictated by the $w_i = 1/(f_i^{\max} - f_i^{\min})$ term, as shown in Figure 2. The local search is terminated if any of the following conditions is met: (i) a maximum of 25 iterations is elapsed (here, maximum iterations are fixed to prevent excessive function evaluations during initial stages of algorithm) (ii) a KKT error value of 0.001 is achieved, or (iii) a maximum difference of 10^{-6} in any variable in two successive iterations is achieved.

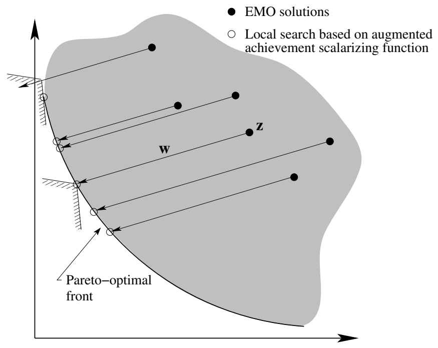

● EMO solutions
○ Local search based on augmented achievement scalarizing function
z
w
Pareto−optimal front

Fig. 2. Proposed local search based on augmented achievement scalarizing function

In this study, we use a probability of local search p_l which periodically increases and drops linearly with generations. Starting from zero at the initial generation, the probability rises to 0.01 in $(0.5N - 1)$ generations (where N is the population size) and drops to zero in $t = 0.5N$ generations. This means that, when $N = 100$, and generation $= (0.5N - 1)$, on an average one solution in the entire population gets modified by the local search. The initial generations have a smaller local search probability, as typically the population is far from the Pareto-optimal front and the local search may mostly produce extreme Pareto-optimal solutions. The probability increases linearly as more solutions may need to be modified using the local search procedure to ensure convergence to the Pareto-optimal front. p_l goes to zero after each period to prevent loss in diversity both during these initial phases and when the population approaches the Pareto-optimal front.

To terminate the hybrid approach, the normal stopping criteria of EMO such as fixed maximum number of generations can be applied. Alternatively, it is time to stop when the local search produces no significant change. Then the local search should be applied to the entire final population. Here, for testing purposes we use a stopping criterion based on the discrepancy (we call it an 'error metric') in objective f_M between obtained solution and corresponding f_M value obtained by substituting other objective values (f_1 to f_{M-1}) in the Pareto-optimal relationship ($f_M = f_M(f_1, \ldots, f_{M-1})$) is calculated for each current non-dominated solutions. If the sum of the square of errors generated by all non-dominated solutions is less than or equal to 0.001, the hybrid approach is terminated. This termination criterion ensures that all obtained solutions are close to the true Pareto-optimal front. It must be noted, that this termination criterion is used only to test the efficacy of our algorithm and cannot be applied to any general problem. To, check the diversity of obtained set of non-dominated solutions, we calculate the hypervolume measure (HV) of the obtained set of points [4]. Thereafter, the normalized difference in hypervolume measure (NDHV = (HV*-HV)/HV*, where HV* is the hypervolume of the true Pareto-optimal front) is computed and compared with the same obtained for the solutions of the original NSGA-II.

Although not obvious, we argue here that the use of a local search procedure within an EMO shall constitute a computationally faster approach than without the use of a local search to EMO procedure. The occasional use of the local search procedure will introduce a few elite solutions in the population. Under EMO operators, these solutions will then get an opportunity to recombine with other population members and exchange variables between them, which may cause more non-Pareto-optimal solutions to come closer to the Pareto-optimal front. This hybrid approach allows the population to converge faster near the Pareto-optimal front.

4 Results and Discussion

We apply the proposed hybrid approach on a number of two to four objective test problems. As a local solver we use SQP from KNITRO [2]. To compare the speed and accuracy of our hybrid approach with the original NSGA-II, both algorithms are terminated when the average error metric value is smaller than 0.001 and the number of function calls needed in each case are recorded. For each problem, we also compute and compare the NDHV.

For this study, we consider four bi-objective test problems (ZDT1, ZDT2, ZDT3 and ZDT4) and two three-objective test problems (DTLZ1 and DTLZ2) and one four-objective problem (DTLZ2). For bi-objective problems, we have 100 population members and for three and four objectives, we have used 200 population members. Crossover probability of 0.9, SBX distribution index [4] of 15, mutation probability of 0.1^{M-1} (reduced probability with number of objectives due to increased maintenance of diversity by crowding distance operator), and mutation distribution index of 20 are used. Table 1 show the best, median

Table 1. Comparison of the number of function calls for the hybrid approach and original NSGA-II. Algorithms are terminated when a fixed level of convergence is achieved.

Test	Original NSGA-II			Hybrid approach		
Problem	Best	Median	Worst	Best	Median	Worst
ZDT1	19,400	20,900	24,600	**4,665**	7,554	8,580
ZDT2	20,600	21,700	23,200	**4,826**	6,351	7,198
ZDT3	21,900	23,500	26,300	**10,736**	16,731	22,137
ZDT4	27,100	34,500	60,300	**6,003**	7,658	14,479
3-DTLZ1	59,800	76,200	97,200	**48,352**	60,323	76,890
3-DTLZ2	38,200	54,000	93,800	**32,611**	61,508	69,650
4-DTLZ2	52,200	68,200	120,800	**45,005**	61,879	114,899

Table 2. Comparison of NDHV for the hybrid approach and original NSGA-II. (smaller value is better).

Test	Original NSGA-II			Hybrid approach		
Problem	Best	Median	Worst	Best	Median	Worst
ZDT1	0.0043	0.0047	0.0054	**0.0034**	0.0042	0.1630
ZDT2	0.0044	0.0053	0.0064	**0.0037**	0.0070	0.0499
ZDT3	0.0012	0.0016	0.0023	**0.0007**	0.0009	0.0010
ZDT4	0.0042	0.0047	0.0055	**0.0037**	0.0106	0.223
3-DTLZ1	0.0196	0.0341	0.0403	**0.0187**	0.0224	0.0296
3-DTLZ2	0.0000	0.0000	0.0000	0.0000	0.0001	0.0030
4-DTLZ2	0.0049	0.0066	0.0086	**0.0048**	0.0067	0.0085

and worst function calls for 10 runs started from 10 identical initial populations for both the hybrid approach and the original NSGA-II procedure. The better algorithm in terms of the smallest number of required function calls is marked in bold face. It is clear that for bi-objective problems the convergence is much faster with the hybrid approach. For more objectives, better results are observed with the hybrid approach, but the difference in the performance seems to reduce with an increase in the number of objectives. We suspect that this behavior is due to the degraded performance of domination-based EMO approaches with an increased number of objectives [4]. Although the hybrid approach did not explicitly introduce a mechanism to maintain diversity except NSGA-II's crowding distance operator, we present NDHV values in Table 2. It is interesting that in most cases the proposed hybrid approach is able to find a well-distributed set of converged points. In the case of ZDT4, relatively higher median and worst values, together with a prescribed error measure of 0.001 and a smaller number of function calls, indicate desired convergence at a faster rate, but at the expense of needed diversity in some simulation runs. Since in most problems, a function evaluation is most time consuming, we stress here on number of function calls, rather than exact computational time.

Other ideas could be considered for ensuring diversity, such as: Firstly, the local search direction can be biased differently for different EMO solutions based on the location of EMO solution and on the undiscovered regions of the Pareto-optimal front. Secondly, the crowding distance operator of NSGA-II can be replaced with a better diversity preserving procedure, such as clustering [13,19] and lastly, instead of using a generational evolutionary optimization approach such as NSGA-II, a steady-state procedure (such as in epsilon-MOEA [6]) may be adopted. In this way, every new solution created by the hybrid approach can be evaluated for its convergence and diversity enhancement properties to the rest of the non-dominated solutions of the EMO population.

We now discuss the working principle of the local search by examining its performance on the ZDT1 and ZDT2 problems. In Figures 3 and 4, we plot the average error metric values versus generation counter for both the approaches. The vertical lines indicate the generations at which at least one local search is executed. It is interesting to observe that soon after the first local search has

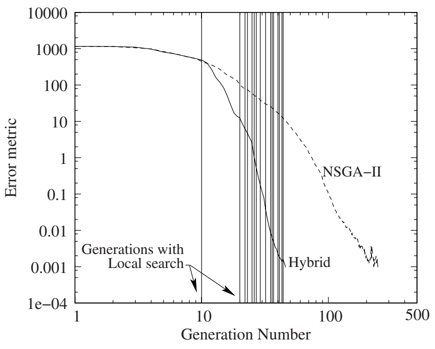


Fig. 3. Average error metric for ZDT1

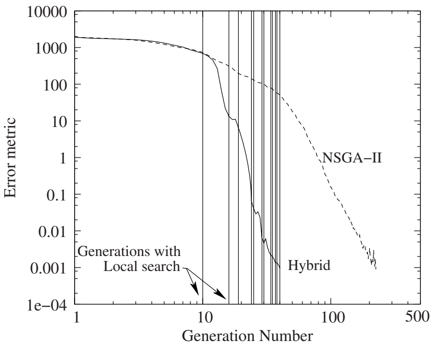


Fig. 4. Average error metric for ZDT2

taken place at generation 10, the population gathered momentum to move towards the Pareto-optimal frontier for both problems. Figure 5 shows all 100 population members at the start of generations 10, 11, 13 and 15. The first local search takes place at generation 10 and a weak Pareto-optimal solution on f_2 axis (marked with a circle) was found (25 iterations of local search were not enough to find proper Pareto optimal solution). The presence of this solution in the population and its subsequent recombination with other population members caused them to come closer to the Pareto-optimal front, thereby providing the speed of convergence. Starting from the same initial population, the original NSGA-II, although maintained a similar error metric value till generation 10, failed to keep pace with the hybrid approach thereafter. Figure 6 shows the populations at the above generations to demonstrate the slow nature of convergence of the original NSGA-II. A similar phenomenon is observed with test problem ZDT2 (Figure 4).

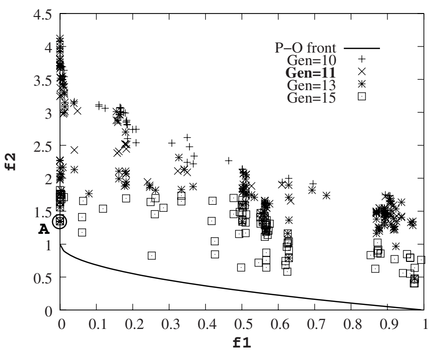


Fig. 5. Populations approach the Pareto-optimal front faster in the hybrid approach

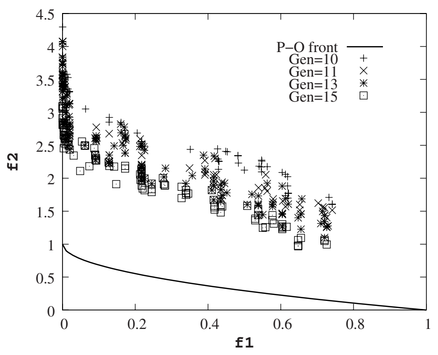


Fig. 6. Populations approach the Pareto-optimal front slowly in the original NSGA-II approach

5 Conclusions

In this paper, we have argued that an efficient implementation of a local search procedure in an EMO algorithm is not straightforward. To take advantage of fast and accurate convergence to Pareto-optimal solutions, EMO algorithms must use a directed and provable local search procedure. In this study, we have suggested the use of an augmented achievement scalarizing function to be solved with a local search method. The local search procedure has been implemented as an additional operator and applied to EMO populations with a varying probability. On a number of standard test problems involving two to four objectives, we have observed that our proposed hybrid approach with NSGA-II is computationally faster than the original NSGA-II procedure in finding solutions which are close to the Pareto-optimal front.

Achieving convergence of solutions to the Pareto front is just half of our quest, as diversity of obtained solutions is also vital. Although in this study we rely on the NSGA-II crowding distance operator for diversity preservation, the faster and accurate convergence achieved with the proposed local search is now ready to be coupled with a more efficient diversity ensuring operator (currently under study). However the present study has clearly shown the advantage and potential of hybridizing local search in a fast and accurate computation of Pareto-optimal solutions.

Acknowledgements

The support by the Academy of Finland (grant 118319), the Foundation of Helsinki School Economics and the Jenny and Antti Wihuri Foundation (Finland) is highly appreciated.

References

1. Belton, V., Stewart, T.: Multiple Criteria Decision Analysis: An integrated Approach. Kluwer, Dordrecht (2002)
2. Byrd, R.H., Nocedal, J., Waltz, R.A.: KNITRO: An integrated package for nonlinear optimization. In: di Pillo, G., Roma, M. (eds.) Large-Scale Nonlinear Optimization, pp. 35–59. Springer, Heidelberg (2006)
3. Coello Coello, C.A., VanVeldhuizen, D.A., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
4. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
6. Deb, K., Mohan, M., Mishra, S.: Towards a quick computation of well-spread Pareto-optimal solutions. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 222–236. Springer, Heidelberg (2003)

7. Goel, T., Deb, K.: Hybrid methods for multi-objective evolutionary algorithms. In: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), pp. 188–192 (2002)
8. Harada, K., Sakuma, J., Kobayashi, S.: Local search for multiobjective function optimization: Pareto descent method. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 659–666 (2006)
9. Ishibuchi, H., Narukawa, K.: Some issues on the implementation of local search in evolutionary multiobjective optimization. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3102, pp. 1246–1258. Springer, Heidelberg (2004)
10. Jaskiewicz, A.: Genetic local search for multiple objective combinatorial optimization. European Journal of Operational Research 371(1), 50–71 (2002)
11. Knowles, J.D., Corne, D.: M-PAES: A memetic algorithm for multiobjective optimization. In: Proceedings of Congress on Evolutionary Computation (CEC 2000), pp. 325–332 (2000)
12. Kuhn, H.W., Tucker, A.W.: Nonlinear Programming. In: Neyman, J. (ed.) Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, pp. 481–492. University of California Press, Berkeley (1951)
13. Kukkonen, S., Deb, K.: A fast and effective method for pruning of non-dominated solutions in many-objective problems. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN IX 2006. LNCS, vol. 4193, pp. 553–562. Springer, Heidelberg (2006)
14. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation 10(3), 263–282 (2002)
15. Laumanns, M., Thiele, L., Zitzler, E., Welzl, E., Deb, K.: Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 44–53. Springer, Heidelberg (2002)
16. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
17. Murata, T., Nozawa, H., Tsujimura, Y., Gen, M., Ishinuchi, H.: Effect of local search on the performance of celluar multi-objective genetic algorithms for designing fuzzy rule based classification systems. In: Proceeding of the Congress on Evolutionary Computation (CEC 2002), pp. 663–668 (2002)
18. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Applications, pp. 468–486. Springer, Berlin (1980)
19. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., Tsahalis, D.T., Périaux, J., Papailiou, K.D., Fogarty, T. (eds.) Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, International Center for Numerical Methods in Engineering (CIMNE), pp. 95–100 (2001)

A Convergence Criterion for Multiobjective Evolutionary Algorithms Based on Systematic Statistical Testing

Heike Trautmann[1], Uwe Ligges[2], Jörn Mehnen[3], and Mike Preuss[4]

[1] Department of Statistics, Technische Universität Dortmund
trautmann@statistik.uni-dortmund.de
[2] Department of Statistics, Technische Universität Dortmund
ligges@statistik.uni-dortmund.de
[3] Decision Engineering Centre, Cranfield University, UK
j.mehnen@cranfield.ac.uk
[4] Chair of Algorithm Engineering, Technische Universität Dortmund
mike.preuss@uni-dortmund.de

Abstract. A systematic approach for determining the generation number at which a specific Multi-Objective Evolutionary Algorithm (MOEA) has converged for a given optimization problem is introduced. Convergence is measured by the performance indicators Generational Distance, Spread and Hypervolume. The stochastic nature of the MOEA is taken into account by repeated runs per generation number which results in a highly robust procedure. For each generation number the MOEA is repeated a fixed number of times, and the Kolmogorow-Smirnov-Test is used in order to decide if a significant change in performance is gained in comparison to preceding generations. A comparison of different MOEAs on a problem with respect to necessary generation numbers becomes possible, and the understanding of the algorithm's behaviour is supported by analysing the development of the indicator values. The procedure is illustrated by means of standard test problems.

1 Introduction

Convergence properties for Multi-Objective Evolutionary Algorithms (MOEA) are an equally important issue as for single objective optimization. The question when to stop a stochastic search algorithm depends on practical as well as on technical decisions. Due to the fact that MOEAs are a fairly recent phenomenon there do not exist many mathematical convergence theories yet. Some of the current theories state back to single-objective theory (Deb 2004). Rudolph and Agapie (2000) and Rudolph (2001) proved that MOEAs with elitism and positive variation kernel can have the property of converging to the true Pareto front in finite number of function evaluations in finite search space problems. Further rigorous results are available for $t \to \infty$ (Hanne 1999). Laumanns et al. (2003, 2002) provided results on theoretical as well as empirical convergence properties for ϵ-MOEAs introducing additional spread properties. Van Veldhuizen and Lamont (1998a) derive results on sufficient conditions of convergence.

In contrast to offline convergence analysis also online termination criteria for an MOEA exist, i.e. the MOEA is started for a single run, and it is stopped once a specific termination criterion has been met. In practice it is generally difficult to find a good termination criterion for MOEAs without sufficient a-priori knowledge about the optimization problem at hand. The most frequently used termination criterion is the maximum number of generations or execution time. An alternative is to measure the difference of improvements during a certain time interval. If the improvement is smaller than a certain threshold the termination criterion holds. The problem here lies in the determination of the threshold. Furthermore, this criterion may also be misleading in cases of functions with very small inclinations. Another criterion is to stop after a certain quality of a solution is reached. This leaves the problem of choosing a proper quality limit that allows for a finite termination of the algorithm. Rudenko and Schoenauer (2004) mention various online termination criteria for elite MOEAs. They discuss e.g. disappearance of all dominated individuals or deterioration of the number of newly produced non-dominated individuals. They propose a technique for determining stagnation using a stability measure of the crowding distance (Deb 2002). Deb and Jain (2002) investigate so-called running performance metrics for convergence and diversity of solutions to be monitored in the course of the algorithm.

Especially for problems with yet unknown characteristics a systematic analysis of the required run length of the MOEA becomes necessary as no sufficient a-priori-knowledge is available for choosing the desired solution quality. Termination criteria often are heuristical procedures with the disadvantage of being statistically unrobust as only a single run of the algorithm is taken into account.

In this paper a systematic offline convergence analysis of MOEA behaviour with respect to multiple performance indicators is suggested (Testing-based Runlength Detection (TRD)). Here Generational Distance (GD, Van Veldhuizen and Lamont 1998), Spread (Deb 2002) and Hypervolume (Zitzler and Thiele 1999) were chosen exemplary. For each generation number in a predefined (preliminary) interval the MOEA is applied m times resulting in m values of the performance indicators. This makes the procedure very robust as the stochastic nature of the MOEA is addressed.

Subsequently a Kolmogorow-Smirnov-Test (Sheskin 2000) is applied for each indicator in order to check if the distribution of the indicator at a specific generation number significantly differs from the distribution of the indicator values obtained at the five previous generations. The procedure stops in case the p-value of the test exceeds the significance level for three successive generations, and the indicator-specific optimal generation number is determined. The overall optimal generation number then comes out as the maximum of the indicator-specific optimal numbers. In case the preliminary upper generation limit has not been high enough it is redefined and the procedure is restarted beginning at the previous upper generation limit. Thus no computational recources are wasted.

By the proposed procedure it becomes possible to compare different MOEAs on an optimization problem in a systematic way with respect to the required generation numbers. Ideally the remaining MOEA-parameters should be adapted or tuned upfront for the given problem to allow highest possible algorithm performance. The method also ideally suits problems which have to be optimised repeatedly with the same or slightly different set-up in order to have a guideline when to terminate the MOEA based on systematic and robust analysis. Certainly it can only be considered for optimization problems which require high accuracy on the one hand and allow time for such a systematic and computationally intensive approach on the other hand. This is the case for recurring optimization problems in particular, for which it can be analysed how the MOEA run length interacts with the remaining MOEA parameters.

In Section 2 a detailed description of the proposed procedure is given illustrated by a flow chart. The results of exemplary applications to standard test problems are summarized in Section 3 for NSGA-II (Deb 2002) applied to the Binh- (Section 3.1), the ZDT1- (Section 3.2) and the Fonseca-problem (Section 3.3). Finally conclusions are given in Section 4.

2 A Convergence Criterion for MOEAs

For a given multiobjective optimization problem a simulation is carried out following the algorithm steps in Figure 1. A multiobjective evolutionary algorithm is selected and all parameters except the number of generations are set to a constant level.

For each generation number $G^* \in [G_L, G_U]$ with a predefined step-width S the MOEA algorithm is applied a fixed number of times m always restarting from the beginning, where G_L and G_U are the lower and (preliminary) upper limits of the generation numbers. The parameter m should not be too small to ensure a sound statistical analysis. For the experiments in Section 3 $m = 50$ was used. This procedure results in m Pareto fronts for generation G^*. Ideally a step-width $S = 1$ should be chosen. But as the simulations require much CPU time and difficult test problems need a very high number of generations the step-width S helps to keep the computational effort at an acceptable level. The “true” optimal generation number then is within the interval $[G_{opt} - (S-1), G_{opt} + (S-1)]$.

For all G^* three MOEA performance indicators are chosen in order to determine the solution quality of the MOEA-algorithm. As all the indicators are sensitive regarding the scaling of the objectives all solutions are normalized to the interval $[0, 1]$ before starting the performance analysis. Generational distance is used, measuring the distance of the elements of the last non-dominated objective vectors (the estimated Pareto fronts) of the different algorithms to those in the “true” Pareto optimal front. It is defined as $GD := (\sqrt{\sum_{i=1}^{N} d_i^2})/N$, where N is the number of solutions in the last non-dominated front found by the MOEA, and d_i is the euclidean distance between each of these objective vectors and the nearest objective vector of the true Pareto front. In case GD equals zero all the solutions belong to the true Pareto front.

As a diversity measure the Spread Indicator (SP) introduced by Deb (2002) is used:

$$SP := \left(\sum_{m=1}^{M} d_m + \sum_{i=1}^{N-1} |d_i - \bar{d}| \right) / \left(\sum_{m=1}^{M} d_m + (N-1)\bar{d} \right). \quad (1)$$

The values d_m are the euclidean distances between the extreme solutions of the true Pareto optimal front and the boundary solutions of the last non-dominated set produced by the MOEA corresponding to the m^{th} objective function. M is the total number of

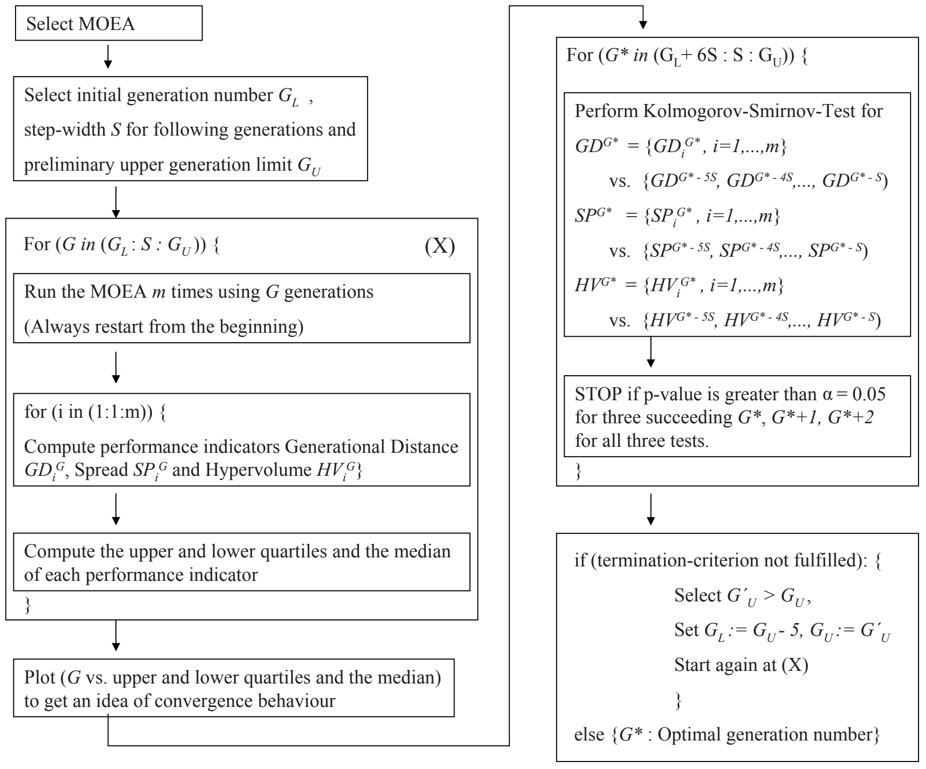

Select MOEA
Select initial generation number G_L, step-width S for following generations and preliminary upper generation limit G_U
For (G *in* ($G_L : S : G_U$)) { (X)
Run the MOEA m times using G generations (Always restart from the beginning)
for (i in (1:1:m)) {
Compute performance indicators Generational Distance GD_i^G, Spread SP_i^G and Hypervolume HV_i^G}
Compute the upper and lower quartiles and the median of each performance indicator
}
Plot (G vs. upper and lower quartiles and the median) to get an idea of convergence behaviour
For (G^* *in* (G_L+ 6S : S : G_U)) {
Perform Kolmogorov-Smirnov-Test for
$GD^{G^*} = \{GD_i^{G^*}, i=1,...,m\}$ vs. $\{GD^{G^*-5S}, GD^{G^*-4S},..., GD^{G^*-S}\}$
$SP^{G^*} = \{SP_i^{G^*}, i=1,...,m\}$ vs. $\{SP^{G^*-5S}, SP^{G^*-4S},..., SP^{G^*-S}\}$
$HV^{G^*} = \{HV_i^{G^*}, i=1,...,m\}$ vs. $\{HV^{G^*-5S}, HV^{G^*-4S},..., HV^{G^*-S}\}$
STOP if p-value is greater than $\alpha = 0.05$ for three succeeding G^*, G^*+1, G^*+2 for all three tests.
}
if (termination-criterion not fulfilled): {
Select $G'_U > G_U$,
Set $G_L := G_U - 5$, $G_U := G'_U$
Start again at (X)
}
else {G^* : Optimal generation number}

Fig. 1. Procedure for selecting the optimal number of generations for a given multiobjective optimization problem (Testing-based Runlength Detection (TRD))

objective functions. The parameter d_i represents the euclidean distance between neighbouring solutions in the last non-dominated set of solutions measured in the objective space, and $\bar{d}$ is the arithmetic average of these distances. The target value of zero for SP indicates a perfect equally distributed spread of the obtained algorithm solutions.

Hypervolume (HV, Zitzler and Thiele 1999) as the third performance criterion reflects the volume in the objective space covered by the members $\mathbf{p}_i (i = 1, \ldots, N)$ of a non-dominated set *ND* of solutions. It is defined relative to an "anti-optimal" reference point R, the worst possible point in the objective space. This point is usually not known and has to be chosen carefully (Knowles and Corne (2002)). It is mostly approximated by the worst objective values in each dimension from any of the calculated fronts in the course of the algorithm. Then the HV is the union of the hypercuboids (bounded by R) in the Lebesgue measure Λ which are weakly dominated by the vectors $\mathbf{p}_i$:

$$HV(ND, R) = \Lambda\left(\left\{\bigcup h(\mathbf{p}_i) | \, \mathbf{p}_i \in ND, i = 1, \ldots, N\right\}\right), \tag{2}$$

$$h(\mathbf{p}_i) = [p_{i1}, R_1] \times [p_{i2}, R_2] \times \ldots \times [p_{iM}, R_M]. \tag{3}$$

Thus the larger the hypervolume the wider is the range of Pareto optimal solutions. Therefore hypervolume has to be maximized.

In order to get an idea about the convergence behaviour of the three indicators for each G^* the upper and lower quartiles as well as the median of the indicator values are computed separately for each indicator and three plots are generated which plot these statistics against the number of generations (see Figure 4 for an example). This relates to the concept of Generalized Runtime Distributions (Hoos and Stützle 2004). By analyzing the plots a first visual analysis for determining the optimal number of generations becomes possible. The MOEA has converged at generation G^* with respect to an individual performance indicator if the median, lower and upper quartile of the indicator values at G^* do not differ significantly from the respective values obtained at adjacent generation steps.

In order to perform a statistically sound analysis however, a sequential statistical test procedure is suggested as follows. Starting at generation $G_L + 6$ separately for each indicator the distribution of the m indicator values at generation G^* is compared to the distribution of all $5 \cdot m$ indicator values of the five previous generation steps. The value of five has been empirically set based on extensive simulation studies on standard test problems. It turned out that smaller values do not allow reliable convergence statements whereas higher values may lead to a too strict procedure, and thus to much higher generation numbers than desired compared to the results of the visual analysis of the performance plots.

The two-sample Kolmogorow-Smirnov (K-S) -Test (Sheskin 2000) is applied with the null-hypothesis that the respective distributions are identical. Thus it is tested if a significant change in performance is obtained at generation G^* compared to the five preceding generation steps.

The K-S-test was chosen as it is a non-parametric test. As the proposed procedure should be generally applicable for all possible multiobjective optimization problems no distribution assumption can be made for the m performance indicator values at each generation. Furthermore, it has a general alternative, i.e. it is sensitive to any kind of distributional difference (location, dispersion, skewness, kurtosis,...) (Siegel and Castellan (1988)). It is desired to check whether the distributions are different in any kind of structure. The K-S-test is easily applicable and tables with critical values have been provided. It proved to be superior to other well-known test procedures, e.g. based on iterations of indicator values (Wald and Wolfowitz 1940) or to the test by Katzenbeisser and Hackl (1986).

The two-sample Kolmogorow-Smirnow-Test relies on the empirical distribution functions of the two samples which are an unbiased estimator for the unknown distribution functions of the underlying populations. Let $I^{G^*} := \{I_i^{G^*}, i = 1, \ldots, m\}$ be the m realizations of a performance indicator I at generation G^* understood as a sample of the population with distribution function $F^{I^{G^*}}$. Furthermore, $I^{G_5^*} := \{I^{G^*-5S}, \ldots, I^{G^*-S}\}$ is defined as the set of $5m$ values of the performance indicator I obtained at the five generations preceding G^*. $I^{G_5^*}$ represents a sample of the population with distribution function $H^{I^{G_5^*}}$. The related empirical distribution functions are $F_m^{I^{G^*}}$ and $H_{5m}^{I^{G_5^*}}$, the index referring to the sample size:

$$F_m^{I^{G^*}}(z) = \begin{cases} 0, & z < I_{(1)}^{G^*} \\ i/m, & I_{(1)}^{G^*} \leq z < I_{(i+1)}^{G^*} \quad \text{for } i = 1, \ldots, m-1 \\ 1, & z \geq I_{(m)}^{G^*} \end{cases} \tag{4}$$

The set $\{I_{(1)}^{G^*}, \ldots, I_{(m)}^{G^*}\}$ is formed by the values of I^{G^*} in ascending order. $H_{5m}^{I^{G_5^*}}$ is defined in a straightforward manner:

$$H_{5m}^{I^{G_5^*}}(z) = \begin{cases} 0, & z < I_{(1)}^{G_5^*} \\ i/(5m), & I_{(1)}^{G_5^*} \leq z < I_{(i+1)}^{G_5^*} \quad \text{for } i = 1, \ldots, 5m-1 \\ 1, & z \geq I_{(5m)}^{G_5^*} \end{cases} \tag{5}$$

The K-S-test assumes independence of the two samples. This condition is fulfilled due to the fact that the MOEA is started from the beginning for each generation number G^*, which is repeated m times. It is no online termination criterion, i.e. it is not the case that every m generation the current non-dominated set is analysed. Thus the objective values in the last non-dominated front at generation G^* are not dependent on the respective objective values at generation $(G^* - 1)$.

The test problem then can be stated as follows :

$$H_0 : F^{I^{G^*}}(z) = H^{I^{G_5^*}}(z) \qquad \forall z \in \mathbb{R}, \tag{6}$$

$$H_1 : F^{I^{G^*}}(z) \neq H^{I^{G_5^*}}(z) \qquad \text{for at least one } z \in \mathbb{R}. \tag{7}$$

The test statistic $K_{m,5m}$ (8) of the K-S-test relies on the maximum absolute difference of the respective empirical distribution functions,

$$K_{m,5m} = \max_z |F_m^{I^{G^*}}(z) - H_{5m}^{I^{G_5^*}}(z)|, \tag{8}$$

and H_0 is rejected if $K_{m,5m} > k_{1-\alpha}$. It can be shown that $K_{m,5m}$ is non-parametric, i.e. does not depend on the concrete z values at hand. For $m, n \geq 20$ and chosen significance level $\alpha = 0.05$ the critical value $k_{1-\alpha}$ can be approximated by $1.36\sqrt{\frac{m+(5m)}{m\cdot(5m)}}$ (Sheskin 2000). The significance level $\alpha = 0.05$ was chosen as it is a standard value for statistical testing, and as it led to very good and reasonable simulation results (Section 3).

Opposed to the standard statistical test approach however it is not of interest if H_0 is rejected. Instead one is waiting for the specific G^* that allows to conclude that H_0 cannot be rejected *any more*. With respect to the performance indicator at hand ($I \in \{GD, SP, HV\}$) the optimal generation number G_{opt} is then selected as the generation number from where the p-value of the K-S-test exceeds 0.05 for three consecutive generation steps. The value of three was determined based on simulation studies on standard test problems (see Section 3). Smaller values do not produce stable enough results, i.e. fluctuating p-values may occur. Convergence decisions are not reliable enough, i.e. the procedure tends to stop too early. Especially for multimodal problems it could happen that the MOEA gets completely stuck in a local optimum. Higher values on the other hand tend to cause a too strong convergence criterion, especially if S is quite large.

As the test procedure is applied separately for all three performance indicators a multiple test problem (Miller 1981) results at each generation number resulting in a global significance level of 0.15 at each G^*. This implies a probability of 15 % that H_0 is rejected although it actually holds.

The overall optimal generation number for the multiobjective optimization problem at hand is chosen as the maximum of the three individual optimal generation numbers of the indicators. In case the upper generation limit G_U that has been chosen upfront turns out not to be high enough to meet the convergence-criterion of the algorithm, a new upper limit $G'_U > G_U$ has to be defined, and the procedure starts from the beginning by setting $G_L := G_U - 5$ and $G_U := G'_U$. However, one should be aware of the fact that there may exist problems for which the MOEA will not find a good approximation of the true Pareto front, regardless the number of generations. In our simulation the Deb(3)-problem (Collette and Siarry 2003) turned out to be such an example. This problem is difficult due to the change of expected PF-shape from concave to convex. The percentage in which the algorithm failed, i.e. did not succeed in finding a convex front was not stable at all over the m runs, even not for very high generation numbers. In such cases reasonable considerations are necessary to select a suitable generation number based on the plots mentioned above.

Although three specific performance indicators were chosen for the analysis the proposed procedure is independent of the type and number of performance indicators. One can easily replace e.g. GD by another criterion or even add another performance criterion for the analysis. However, the significance levels of the K-S-test for the individual indicators then have to be adjusted with respect to the desired global significance level. In case the true Pareto front is not known upfront indicators have to be chosen which are independent from this a-priori-knowledge. It is even possible to solely focus on Hypervolume as an appropriate indicator which would remove the multiplicity of the test problem.

For completely unknown problems the appropriate choice of the parameters G_L, G_U, and S could be difficult. The recommendation in this case is to use small numbers for G_L and S as the problem complexity is not known. The choice of G_U is not crucial as it can be adapted in the course of the procedure without any computational loss as explained above.

3 Simulations

The proposed procedure was tested by means of three standard benchmark test problems, the Binh-, the ZDT1- and the Fonseca-problem. The performance indicators have been determined using JMetal-Tools (http://mallba10.lcc.uma.es/wiki/index.php/Tools). A real-coded Matlab-implementation of NSGA-II (Deb 2002) was used with parameter settings as typically suggested for the respective test functions (Deb 2004). The population sizes will be given with the fitness functions. The pool size is half the population size, and the distribution indices for crossover and mutation operators are $\mu = 20$ and $\mu_m = 20$, respectively. The number of repeated runs for each generation number was set to $m = 50$.

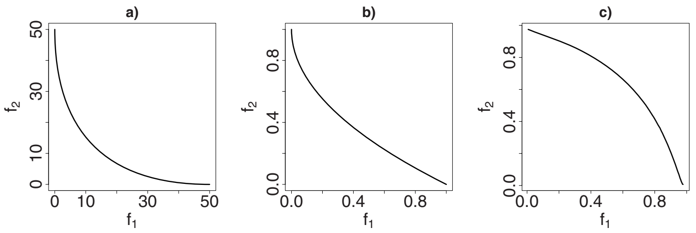


Fig. 2. Pareto-fronts of the a) Binh- , b) ZDT1- and c) Fonseca-problem

In most test problems the SP indicator showed earlier convergence than GD implying that the NSGA-II tends to first build up a well-spread non-dominated set and then tries to shift it in the direction of the true Pareto front as accurate as possible. Furthermore, the plots reveal that the improvement in the HV indicator is only large from lower generation numbers to medium-sized ones. Visual inspection of the graphs indicate that convergence seems to occur quite early. However, the distribution function of the HV

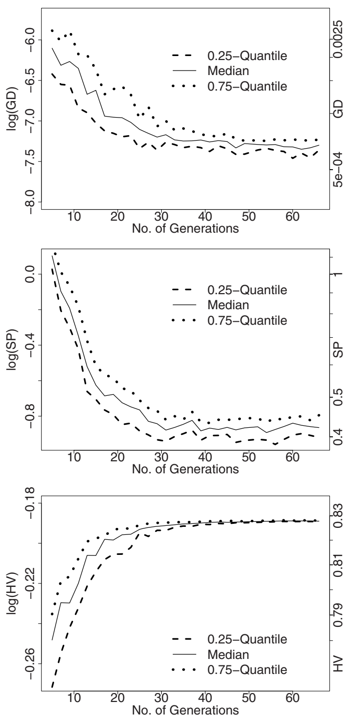


Fig. 3. NSGA-II applied to the Binh-problem; GD, SP and HV are analysed

Table 1. P-values of K-S-Test for Binh-Problem

Gen	p (GD)	p (SP)	p (HV)
15	0.0006	0.0000	0.0000
17	0.0000	0.0000	0.0000
19	0.0011	0.0000	0.0000
21	0.0544	0.0000	0.0001
23	0.0544	0.0001	0.0024
25	0.0003	0.0001	0.0000
27	0.0119	0.0001	0.0003
29	0.0024	0.0013	0.0001
31	0.2124	0.0000	0.0072
33	0.0009	0.0263	0.0005
35	**0.1190**	0.0473	0.0029
37	0.0926	**0.1344**	0.0119
39	0.0544	0.0926	0.0004
41	0.0624	0.2124	0.0225
43	0.3210	0.7583	0.0060
45	0.9822	0.9525	0.0410
47	0.0035	0.5021	0.0029
49	0.0410	0.5435	0.3210
51	0.4240	0.5021	0.0263
53	0.2365	0.2626	0.0042
55	0.6728	0.5021	0.0042
57	0.2365	0.6728	0.0011
59	0.1699	0.0926	0.0263
61	0.0713	0.3210	**0.1513**
63	0.2365	0.9304	0.6728
65	0.1052	0.6728	0.1190

values of the m runs becomes very steep for higher generation numbers so that even a small number of different solutions affects the shape of the distribution drastically. Thus the HV convergence criterion is always met much later than visually expected.

3.1 Binh-Problem

Figure 2 a) shows the Pareto-front for the Binh-problem (Binh 1999, Collette and Siarry 2003), which is a connected convex curve. The generation limits G_L and G_U are set to 5 and 65 using a step width S equal to 2 whereas the population size equals 50. The true Pareto front can be computed using the analytical formula $f_2(x) = 2 \cdot (\sqrt{f_1(x)/2} - 5)^2$ with $f_1(x) \in [0, 50]$. It has been approximated using step width 0.1 in the interval $f_1(x) \in [0, 10]$ and step width 0.5 in the interval $f_1(x) \in]10, 50]$.

Figure 4 visualizes the median as well as upper and lower quartiles of the performance indicators in relation to the number of generations. GD converges at first which is reflected by the results of the test procedure in Table 1. The optimal generation numbers are 35 with respect to GD, 37 with respect to SP and 61 for HV leading to an overall optimal generation number of 61.

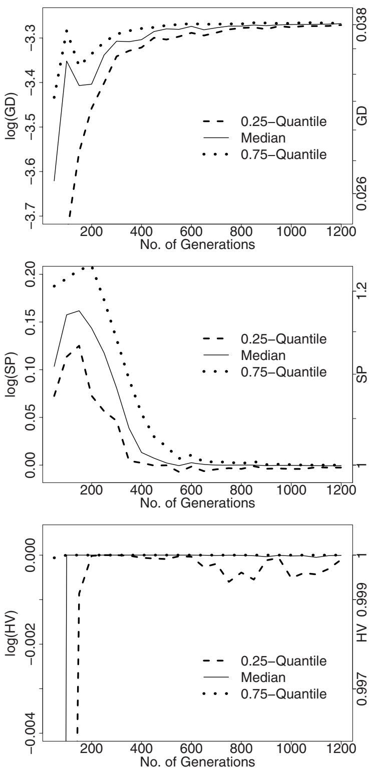


Fig. 4. NSGA-II applied to the ZDT1-problem; GD, SP and HV are analysed

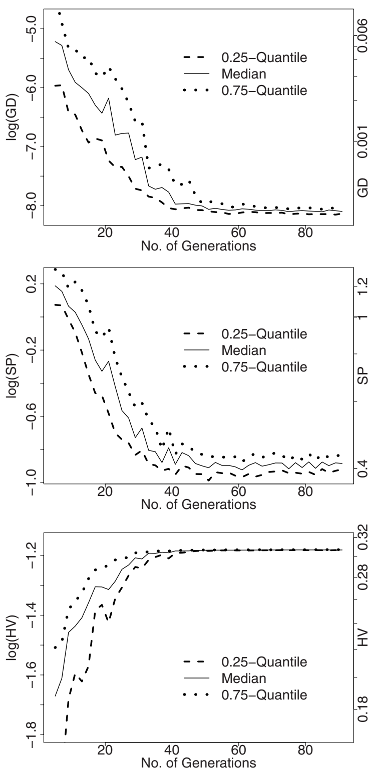


Fig. 5. NSGA-II applied to the Fonseca-problem; GD, SP and HV are analysed

3.2 ZDT1-Problem

The ZDT1-problem (Zitzler 1999) has 30 decision variables. Its Pareto front is a convex and connected curve (Figure 2 b)) and can be analytically determined by $f_2(x) = 1 - \sqrt{f_1(x)}$. The true front has been approximated by this formula for $f_1 \in [0, 1]$ with a sequence step of 0.05.

As the ZDT1-problem is known to be difficult to solve, even by using generation numbers greater than 1000 (Zitzler 1999) a step width $S = 50$ is adequate with regard to the computational effort of the simulations. G_L was chosen as 50 and the upper generation limit G_U as 1200. The population size equals 200 as in the simulations performed by Zitzler (1999). Figure 4 reflects that SP and HV behave rather similar regarding the convergence behaviour, suggesting optimal generation numbers of 700 and 650 (Table 3). The p-values of the K-S-test for GD however do not exceed the significance level of 0.05 for three succeeding generation steps until 1000 generations which therefore is the overall optimal generation number. Not according to expectation the Generational Distance Indicator does not have a downward but an upward tendency. This is due to the fact that for small generation numbers the range of solution quality differs very much. Also some outliers with very good GD values come up depending on the values of the initial front. These solutions however often behave very poor with regard to SP. With increasing generation numbers the MOEA behaviour then becomes stable regarding a specific GD level.

Table 2. P-values of K-S-Test for ZDT1-Problem

Gen	p (GD)	p (SP)	p (HV)
300	0.0000	0.0005	0.0085
350	0.0000	0.0000	0.5860
400	0.0000	0.0000	0.0926
450	0.0000	0.0000	0.0225
500	0.0000	0.0000	0.0225
550	0.0060	0.0000	0.2908
600	0.0004	0.0119	0.0410
650	0.3533	0.0060	**0.1052**
700	0.0473	**0.0814**	0.0713
750	0.0473	0.0814	0.3533
800	0.0009	0.2908	0.1344
850	0.0072	0.2908	0.1902
900	0.2908	0.0306	0.4240
950	0.0003	0.6292	0.2908
1000	**0.5435**	0.3533	0.7990
1050	0.1513	0.0354	0.9525
1100	0.3877	0.0624	0.1190
1150	0.4240	0.2908	0.9304
1200	0.0193	0.3877	0.5021

Table 3. P-values of K-S-Test for Fonseca-Problem

Gen	p (GD)	p (SP)	p (HV)
15	0.0140	0.0000	0.0001
17	0.0020	0.0000	0.0000
19	0.0624	0.0000	0.0000
21	0.0544	0.0263	0.0713
23	0.0060	0.0009	0.0140
25	0.1190	0.0000	0.0011
27	0.0193	0.0000	0.0000
29	0.0050	0.0000	0.0000
31	0.0263	0.0050	0.0035
33	0.0000	0.0001	0.0000
35	0.0001	0.0029	0.0000
37	0.0410	0.0000	0.0000
39	0.0165	0.3533	0.3877
41	0.0000	0.0001	0.0001
43	0.0035	0.1052	0.0050
45	0.0410	0.6728	0.0060
47	0.0193	0.0814	0.0000
49	0.0007	0.0193	0.0000
51	0.0050	0.0306	0.0006
53	0.0926	**0.0926**	0.0119
55	0.0225	0.2626	0.0024
57	0.0024	0.9697	0.0072
59	**0.1513**	0.5021	**0.4240**
61	0.9036	0.1344	0.6728
63	0.7160	0.8372	0.7583
65	0.3210	0.2365	0.7583
67	0.3877	0.9905	0.1902

3.3 Fonseca-Problem

The Pareto front of the Fonseca-problem (Fonseca and Fleming 1995) is a concave and connected curve (Figure 2 c)). The interval $[G_L, G_U]$ was chosen as $[15, 67]$ using a step width S of 2, and the population size of NSGA-II was set to 100. The true Pareto front has been approximated via NSGA-II using a population size of 100 and 200 generations. While Figure 5 suggests an optimal generation number of around 40 the K-S-tests determines the individual optimal generation numbers as 53 (SP) and 59 (GD, HV). Therefore 59 is the overall optimal generation number.

4 Conclusions

Specifying the optimal number of generations for a multiobjective evolutionary algorithm is not an easy task. On the one hand no computational resources should be wasted, but on the other hand the results have to be accurate enough. Furthermore, the optimal number differs for each individual multiobjective optimization problem.

This paper provides a systematic procedure based on statistical testing called Testing-based Runlength Detection (TRD). In contrast to an online termination criterion the evolutionary algorithm is applied m times sequentially for each generation number up to a preliminary upper limit. Per run three MOEA performance indicators are computed (Generational Distance, Spread and Hypervolume). At each generation number a Kolmogorow-Smirnov-Test is used to test if the distribution of the indicators over the m runs of the algorithm is significantly different from the distribution of the indicator values of the five previous generation numbers. This accounts for the stochastic nature of the MOEA resulting in a very robust analysis of the algorithm's performance behaviour. The generation where the p-value exceeds the significance level for three consecutive generation steps is the optimal generation number for a performance indicator. By applying the maximum to all indicator-specific optimal numbers the procedure yields the overall optimal generation number.

Simulations on standard test cases show the easy applicability and high efficiency of the proposed method. However, it is computationally intensive and thus designated for problems which require high accuracy and can afford time for detailed and systematic evaluation of the algorithm performance. It is designed for a well-founded and robust comparison between different MOEAs on given test problems with regard to the required generations until convergence. By analysing the behaviour of the three perfomance indicators over time conclusions regarding the MOEA solution strategy can be made. An application to other MOEA run length criteria (e.g. function evaluations) is straightforward once it has been decided on desired intervals and step-widths of the criteria.

References

Binh, T.T.: A Multiobjective Evolutionary Algorithm: The Study Cases. Technical Report, Institute for Automation and Communication, Berleben, Germany (1999)

Collette, Y., Siarry, P.: Multiobjective Optimization, Principles and Case Studies. Springer, Berlin (2003)

Deb, K., Jain, S.: Running Performance Metrics for Evolutionary Multi-Objective Optimisation. In: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), Singapore, pp. 13–20 (2002)

Deb, K., Pratap, A., Agarwal, S.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(8), 182–197 (2002)

Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, New York (2004)

Fonseca, C.M., Fleming, P.J.: Multiobjective Genetic Algorithms Made Easy: Selection, Sharing, and Mating Restriction. In: Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Sheffield, UK, pp. 42–52 (1995)

Hanne, T.: On the convergence of multiobjective evolutionary algorithms. European Journal of Operational Research 117(3), 553–564 (1999)

Hoos, H.H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Elsevier / Morgan Kaufmann, San Francisco (2004)

Katzenbeisser, W., Hackl, P.: An alternative to the Kolmogorov-Smirnov two-sample Test. Communications in Statistics – Theory and Methods 15, 1163–1177 (1986)

Knowles, J., Corne, D.: On Metrics for Comparing Nondominated Sets. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC), Piscataway, New Jersey, pp. 711–716 (2002)

Laumanns, M.: Analysis and Applications of Evolutionary Multiobjective Optimization Algorithms. PhD Thesis, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland (2003)

Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining Convergence and Diversity in Evolutionary Multi-objective Optimization, Evolutionary Computation 10(3), 263-282 (2002)

Miller, R.G.: Simultaneous Statistical Inference, 2nd edn. Series in Statistics. Springer, Berlin (1981)

Rudolph, G., Agapie, A.: Convergence Properties of Some Multi-objective Evolutionary Algorithms. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC), pp. 1010–1016 (2000)

Rudenko, O., Schoenauer, M.: A Steady Performance Stopping Criterion for Pareto-based Evolutionary Algorithms. In: Proceedings of the 6th International Multi-Objective Programming and Goal Programming Conference, Hammamet (Tunesia) (2004)

Rudolph, G.: Self-Adaptive Mutations Lead to Premature Convergence. IEEE Transactions on Evolutionary Computation 5(4), 410–414 (2001)

Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures, 2nd edn. Chapman & Hall, New York (2000)

Siegel, S., Castellan Jr., N.J.: Nonparametric statistics for the behavioral sciences, 2nd edn. McGraw-Hill, New York (1988)

Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective Evolutionary Algorithm Research: A History and Analysis. Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson, TR 98 03 (1998)

Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary computation and convergence to a Pareto front. In: Proceedings of the Third Annual Conference on Genetic Programming, San Francisco, CA, pp. 221–228 (1998a)

Wald, A., Wolfowitz, J.: On a test whether two samples are from the same population. Annals of Mathematical Statistics 11, 147–162 (1940)

Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)

A Proposal to Hybridize Multi-Objective Evolutionary Algorithms with Non-gradient Mathematical Programming Techniques

Saúl Zapotecas Martínez and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
México D.F. 07300, México
saul.zapotecas@gmail.com, ccoello@cs.cinvestav.mx

Abstract. The hybridization of multi-objective evolutionary algorithms (MOEAs) with mathematical programming techniques has gained increasing popularity in the specialized literature in the last few years. However, such hybrids normally rely on the use of gradients and, therefore, normally consume a high number of extra objective function evaluations in order to estimate the gradient information required. The use of direct (nonlinear) optimization techniques has been, however, less common in the specialized literature, although several hybrids of this sort have been proposed for single-objective evolutionary algorithms. This paper proposes a hybridization between a well-known MOEA (the NSGA-II) and two direct search methods (Nelder and Mead's method and the golden section algorithm). The aim of the proposed approach is to combine the global search mechanisms of the evolutionary algorithm with the local search mechanisms provided by the aforementioned mathematical programming techniques, such that a more efficient (i.e., with a lower number of objective function evaluations) approach can be produced.

1 Introduction

The use of evolutionary algorithms for solving multi-objective optimization problems has become very popular in the last 10 years, finding applications in a wide variety of areas [1]. However, one of the current limitations of MOEAs is their computational cost, which turns out to be unaffordable in certain real world applications. The design of hybrid approaches combining MOEAs and mathematical programming techniques is not new (see for example [2]). However, these hybridation schemes normally rely on gradient-based information to guide the search. This may be inappropriate, since estimating such gradients normally requires additional objective function evaluations, which is precisely what we are trying to avoid in computationally expensive problems. Although the use of surrogate models is a possible alternative to deal with such problems (see for example [3]), these approximate models tend to produce accumulated errors that sometimes generate a significant deviation with respect to the original model. In this paper, we propose a new multi-objective hybrid algorithm based on the NSGA-II [4], coupled with two mathematical programming methods: Nelder and Mead's method (which is used for multidimensional optimization) and the golden section (which is used for unidimensional optimization). The aim of this proposed approach is to speed up the convergence of the baseline MOEA, through the introduction of powerful local search engines (based on direct search methods taken from the mathematical programming literature). This sort of hybrid aims at introducing information obtained from mathematical programming techniques (which are deterministic algorithms) to refine the search performed by a global search engine (a genetic algorithm in this case) without having to perform additional objective function evaluations.

2 Basic Concepts

2.1 The Nonlinear Simplex Method

Spendley et al. [5] presented the basic simplex method, which is an efficient sequential optimization method for minimizing real and multidimensional functions. Later on, Nelder and Mead presented an improvement of this method which was called the Nonlinear Simplex Search (NSS) method [6] (also known as the Nelder and Mead method). The convergence towards a minimum value at each iteration of the NSS is conducted by four movements in a geometric shape called *simplex*. A *simplex* or *n-simplex* Δ is a convex hull of a set of $n+1$ affinely independent points Δ_i ($i = \{0, 1, \ldots, n\}$), in some Euclidean space of dimension n. To define the full algorithm, it is necessary to specify four scalar parameters to control the movements performed in the simplex: reflection (α), expansion (γ), contraction (β) and shrinkage (δ). At each iteration, the $n+1$ vertices of the simplex Δ_i represent the solutions evaluated and are sorted according to: $f(\Delta_0) \leq f(\Delta_1) \leq \cdots \leq f(\Delta_n)$. In this way, the movements performed in the simplex by the NSS method are defined as:

1. *Reflection*: $x_r = (1+\alpha)x_c - \alpha\Delta_n$.
2. *Expansion*: $x_e = (1+\alpha\gamma)x_c - \alpha\gamma\Delta_n$.
3. *Contraction*:
   (a) *Outside*: $x_{co} = (1+\alpha\beta)x_c - \alpha\beta\Delta_n$.
   (b) *Inside*: $x_{ci} = (1-\beta)x_c + \beta\Delta_n$.
4. *Shrinkage*: The new vertices of the simplex at the next iteration will be: $\{\Delta_0, v_1, v_2, \ldots, v_n\}$, where $v_j = \Delta_0 + \delta(\Delta_j - \Delta_0)$, for all $j = \{1, 2 \ldots, n\}$.

where x_c is called centroid of the simplex, and is computed as: $x_c = \frac{1}{n}\sum_{i=0}^{n-1} \Delta_i$; Δ_0 and Δ_n are the best and the worst solutions identified within the simplex, respectively. At each iteration, the initial simplex is modified by one of the above movements, according to the following rules:

1. If $f(\Delta_0) \leq f(x_r) \leq f(\Delta_{n-1})$, then $\Delta_n = x_r$.
2. If $f(x_e) < f(x_r) < f(\Delta_0)$, then $\Delta_n = x_e$,
   otherwise $\Delta_n = x_r$.

3. If $f(\Delta_{n-1}) \leq f(x_r) < f(\Delta_n)$ and $f(x_{co}) \leq f(x_r)$, then $\Delta_n = x_{co}$; otherwise, perform a shrinkage.
4. If $f(x_r) \geq f(\Delta_n)$ and $f(x_{ci}) < f(\Delta_n)$, then $\Delta_n = x_{ci}$; otherwise, perform a shrinkage.

We chose Nelder and Mead's method for two reasons: it is a widely used multi-dimensional optimization technique, and there exists previous evidence of success in being hybridized with evolutionary algorithms. However, the use of other (more powerful) mathematical programming techniques (e.g., Powell's conjugate direction method) is left for future work.

2.2 The Golden Section Method

The *golden section*, represented by φ, is a line segment sectioned in two parts according to the *golden ratio*, which refers to the ratio between length and height of a rectangle that is required in order to make it more aesthetically pleasant to our senses. The golden ratio has a value of $\varphi \approx 0.618033$. The golden section search method finds the minima of a function within a certain (given) search interval. This approach is very efficient to optimize unimodal, unidimensional and unconstrained functions, and it is based on the main principle of the *region elimination methods*, which establishes that if we assume a function to be minimized f to be strictly unimodal on the interval $a \leq x \leq b$ with a minimum at x^*, and having two points x_1 and x_2 in this interval, such that $a < x_1 < x_2 < b$, then we can conclude [7]:

1. If $f(x_1) > f(x_2)$, then the minimum of $f(x)$ does not lie in the interval (a, x_1). In other words, $x^* \in (x_1, b)$.
2. If $f(x_1) < f(x_2)$, then the minimum does not lie in the interval (x_2, b). In other words, $x^* \in (a, x_2)$.

By using the golden section, the region to be eliminated at each iteration can be maximized, so that the minimum can be found in a more efficient way (i.e., requiring less iterations). We decided to adopt this approach, because it is the most efficient region elimination method and it is a direct search approach (i.e., does not require derivatives). Although other (more powerful) unidimensional optimization techniques exist, they rely either on polynomial approximations (e.g., quadratic estimation) or gradient-based information (e.g., secant, Newton-Raphson and cubic search) and thus limit the type of functions to be optimized.

2.3 Low-Discrepancy Sequences

A *low-discrepancy sequence* is also called a quasi-random or sub-random sequence, and it offers a high degree of uniformity in comparison with the more common uniformly distributed random numbers. Low-discrepancy sequences are commonly used in Monte Carlo simulations of integrals that do not have a closed-form expression in order to achieve variance reduction. Here, we adopt low-discrepancy sequences to construct the simplex in the Nelder and Mead method. Next, we present the two low-discrepancy sequences that are adopted in this work.

Halton Sequence. The *Halton sequence* [8] is a variation of the *van der Corput sequence* [9], differing only in the representation, since the van der Corput sequence uses binary representation and the Halton sequence adopts a different base for each vector coordinate. The Halton sequence is constructed according to a deterministic method based on number theory. For constructing the i-th vector of the Halton sequence in $\mathbb{R}^n$, the first step is to choose n relatively prime numbers $p_1, p_2, \ldots, p_n$. We consider the representation in base p of i, which takes the $i = a_0 + pa_1 + p^2a_2 + \ldots$ form. Each coordinate of the vector is in $[0, 1]$ and is obtained by:

$$r(i,p) = \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \frac{a_3}{p^4} + \cdots$$

In this way, the i-th vector in the Halton sequence (starting with $i = 0$) is defined as:

$$\langle r(i,p_1), r(i,p_2), \ldots, r(i,p_n) \rangle \tag{1}$$

Hammersley sequence. The *Hammersley sequence* [10] is an adaptation of the Halton sequence, which uses $n - 1$ relatively prime numbers. Starting with $i = 0$, the i-th vector in the Hammersley sequence for a set of k vectors is defined as:

$$\left\langle \frac{i}{k}, r(i,p_1), r(i,p_2), \ldots, r(i,p_{n-1}) \right\rangle \tag{2}$$

In an analogous way, each component of the vector in the Hammersley sequence is in $[0, 1]$.

3 Our Proposed Approach

Our proposed approach is called *Nonlinear Simplex Search Genetic Algorithm* (NSS-GA), and combines the explorative power of a MOEA with the exploitative power of the Nelder and Mead method, which acts as a local search engine. The general scheme of the NSS-GA is detailed in Figure 1.

3.1 Local Search

The general idea of this phase is to intensify the search towards better solutions for each objective function, based on an individual of the population. Genetic traits of the best individuals found for each objective function are reproduced using the evolutionary operators of a genetic algorithm. The main goal of this phase is to obtain the λ set using classical optimization methods. Because the Nelder and Mead algorithm was designed to optimize multidimensional functions, when dealing with unidimensional optimizations, the golden section method is used instead. Thus, the λ set is defined as:

$$\lambda = \lambda_1 \cup \lambda_2 \cup \cdots \lambda_k \cup \varUpsilon$$

where λ_i is a set of the best solutions found for the i-th objective function of the MOP and $\varUpsilon$ is a set of the best solutions found for the aggregating function

```
$n$ = number of decision variables
$\lambda$ = set of locally optimal solutions found by the
local search mechanism
    1. $t = 0$.
    2. Randomly initialize a population $P_t$ of size $N$.
    3. Evaluate the fitness of each individual in $P_t$.
    4. Generate the offspring $Q_t$, applying the selection,
    crossover and mutation operators to $P_t$.
    5. $R_t = P_t \cup Q_t$ (thus, $R_t$ is now of size $2N$).
    6. Assign to $P^*$ the $N$ better individuals from $R_t$
    according to the crowded comparison operator ($\prec_n$).
    7. If ($t \mod \frac{n}{2} = 0$) then:
        i. Get $\lambda$ set according to the exploration phase.
        ii. $R_t^* = P_t^* \cup \lambda$.
        iii. Assign to $P_{t+1}$ the $N$ best
        individuals from $R_t^*$ according to the crowded comparison operator ($\prec_n$).
    Else: $P_{t+1} = P^*$.
    8. $t = t + 1$.
    9. If $t > t_{max}$ stop, else go to step 4.
```

Fig. 1. Main algorithm of our proposed Nonlinear Simplex Search Genetic Algorithm (NSS-GA)

described later in this section. If the i-th objective function to be optimized is unidimensional, the size of λ_i is 1. In this case, the golden section method is adopted to find the minimum of such objective function. Otherwise, if the i-th objective function is multidimensional, then the size of the λ_i set is $n + 1$ and corresponds to all the vertices of the final simplex found by the NSS algorithm. Next, we describe the different components of our local search engine.

Selection Mechanism. In the population P, we choose the individual $x_\Delta \in P$ to optimize its i-th objective:

$$x_\Delta = x_l | x_l = \min_{\forall x_l \in \mathcal{P}^*} \{f_i(x_l)\} \tag{3}$$

where $\mathcal{P}^*$ is a nondominated solutions set within the population P. In other words, the selected individual is the best nondominated solution with respect to the i-th objective function.

Aggregating Function. The vector $\boldsymbol{H} = [f_1^*, f_2^*, \ldots, f_k^*]$, consists of the minimum values f_i^* of the k objective functions in the current generation. We select the individual x_a from the population P, such that we minimize:

$$G(x_a) = \sum_{i=1}^{k} \frac{|\boldsymbol{H}[i] - f_i(x_a)|}{|\boldsymbol{H}[i]|} \tag{4}$$

In this way, the local search minimizes the aggregating function defined by:

$$\psi(x) = ED(\boldsymbol{H}, \boldsymbol{F}(x)) \tag{5}$$

where $\boldsymbol{F}$ is vector of objective functions values of each individual and ED is the Euclidean distance between the $\boldsymbol{F}$ and $\boldsymbol{H}$ vectors. Summarizing, the search first focuses on finding the extremes of the Pareto front (using equation 3). In this phase, we select as many individuals as objectives of the problem. Then, we select one additional individual using equation 4, aiming to reach the "knee" of the Pareto front.

Note that there are functions for which the NSS algorithm becomes inoperable (for example, McKinnon's function [11]). In order to deal with these and other more complex functions, the NSS method has undergone some modifications in the specialized literature (see for example [12,13]). We propose here a new strategy to guide the improvement process towards promising areas during each generation of the hybrid algorithm. Such strategy is described next.

Building the Simplex. The selected solution x_Δ (x_a for the case of the aggregating function) is called "*simplex-head*", which is the first vertex of the n-simplex. The remaining n vertices are created in two phases:

1. *Reducing the Search Domain*: We use a strategy based on genetic analysis of a sample taken from the population. From this sample, we identify the average and standard deviation of the genes (decision variables) in each individual. Based on that information, we define the new search space as:

$$\begin{aligned} low\_bound^j_{new} &= m(P_m(j)) - \sigma(P_m(j)) \\ up\_bound^j_{new} &= m(P_m(j)) + \sigma(P_m(j)) \end{aligned} \tag{6}$$

   where P_m represents the individuals in the sample taken from the population (such individuals are those with the best fitness with respect to the objective function to optimize), $m(P_m(j))$ is the average and $\sigma(P_m(j))$ is the standard deviation in the j-th parameter of the sample P_m. The size of the sample taken in this work is 20% of the total population size.
2. *Build Simplex Vertices*: Once the new search domain has been defined, the remaining vertices are determined using either the Halton or the Hammersley sequence (each has a 50% probability of being selected).

   For both sequences, the components are in $[0, 1]$ and are mapped to the new interval acording to:

$$c' = low\_bound_{new} + c \cdot (up\_bound_{new} - low\_bound_{new})$$

   where c is the component to be mapped to the interval $[0, 1]$ and c' is the component already mapped to the desired interval. Although we do not have a mathematical proof regarding the suitability of this scheme to generate a non-degenerate simplex (i.e., a simplex whose volume is greater than zero), we empirically found that this procedure worked well in the numerous experiments that we performed.

Bounded Variables for NSS. The NSS method was conceived for unbounded domain problems. When dealing with bounded variables, the created vertices can be located outside the allowable bounds after some movements of the NSS method. Luersen et al. [14] proposed a simple strategy to deal with bounded variables, which is the one we adopted in this work:

Let Δ_{new} be the new vertex created by some NSS movement. The j-th component of the vertex is established as:

$$\Delta_i^{(j)} = \begin{cases} low\_bound_j & \text{, if } \Delta_i^{(j)} < low\_bound_j \\ up\_bound_j & \text{, if } \Delta_i^{(j)} > up\_bound_j \\ \Delta_i^{(j)} & \text{, otherwise.} \end{cases} \tag{7}$$

where low_bound_j and up_bound_j are the lower and upper bounds in the j-th parameter, respectively.

However, this strategy can degenerate the simplex. We propose here to rebuild the simplex if it has been degenerated, i.e., if its volume is different from zero. Since we need a procedure to compute the volume of the simplex, we adopt the proposal from [15] for that sake.

Stopping Criteria. Two stopping criteria are adopted in this work. The first criterion imposes convergence towards a vertex better than the worst vertex within the simplex (x_w). This criterion is taken from Lagarias et al. [15]. However, adopting this stopping criterion does not guarantee that the NSS method has an efficient performance. Convergence can be slow and may require a large number of evaluations of the objective function. For this reason, we use a second stopping criterion which consists of defining a convergence threshold ϵ (for the experiments reported in this paper, $\epsilon = 1 \times 10^{-3}$). Thus, the local search is stopped if:

1. It does not generate a vertex better than x_w after performing $(n + 1)$ iterations, or
2. if after performing $2(n+1)$ iterations, the convergence towards a better point is $\leq \epsilon$.

where n is the number of decision variables of the function to be optimized.

4 Comparison of Results

In order to evaluate the performance of the proposed hybrid algorithm, we compare its performance with respect to the NSGA-II [4]. The test problems adopted are five from the ZDT test suite [16] (except from ZDT5, which is a binary test problem). We also adopted two problems from the DTLZ test suite (DTLZ1 and DTLZ2) [17]. The description of these test problems is omitted due to space constraints. We adopted three performance measures to assess our results: Inverted Generational Distance ($\mathcal{IGD}$) [18,1], Spacing ($\mathcal{S}$) [19] and the Coverage Indicator ($\mathcal{CI}$) [16]. Their description is also omitted due to space constraints.

Table 1. Results of $\mathcal{IGD}$ for the NSS-GA and the NSGA-II

Problem	NSS-GA		NSGA-II	
	average	σ	*average*	σ
ZDT1	**0.001149**	0.000598	0.005582	0.000905
ZDT2	**0.002101**	0.001785	0.015385	0.004631
ZDT3	**0.001221**	0.000832	0.004217	0.000798
ZDT4	**0.122063**	0.058813	0.156509	0.051699
ZDT6	**0.008980**	0.004758	0.046699	0.007258
DTLZ1	**0.658650**	0.107311	0.779135	0.168162
DTLZ2	**0.000403**	0.000022	0.000428	0.000024

4.1 Experimental Setup

Our proposal (the NSS-GA) is compared with respect to the baseline algorithm adopted (the NSGA-II). Since our approach does not require any additional parameters for the main search engine (i.e., the NSGA-II), the comparison was done with the same parameter values for both approaches in order to allow a fair comparison. Thus, we adopted the following values: Population size (S_p) = 100, crossover probability (P_c) = 0.9, mutation probability (P_m) = $\frac{1}{N}$, where N is the number of decision variables. The nonlinear simplex search was implemented with: $\alpha = 1, \beta = 2, \gamma = \frac{1}{2}$ and $\delta = \frac{1}{2}$. For each MOP, we performed 30 independent runs with each of the two approaches. The results are presented in Tables 1 to 3. Each table displays both the average and the standard deviation (σ) of each performance measure, for each of the test problems adopted. The best average results obtained in each test function are displayed in **boldface**. Each run is restricted to 4,000 fitness function evaluations, which is a very low value when compared to those adopted by most MOEAs nowadays. From these results, it is evident that our proposed approach (NSS-GA) outperforms the NSGA-II in all the test problems, with respect to both the $\mathcal{IGD}$ (which measures closeness to the true Pareto front) and the $\mathcal{CI}$ performance measure (which determines if the solutions generated by one algorithm dominate the solutions generated by the other). In fact, the low values obtained by our NSS-GA indicate that our approach has practically converged to the true Pareto front, except for ZDT4 and DTLZ1 (we could corroborate this by plotting the results, but such graphs

Table 2. Results of $\mathcal{S}$ for the NSS-GA and NSGA-II

Problem	NSS-GA		NSGA-II	
	average	σ	*average*	σ
ZDT1	**0.014620**	0.005329	0.023731	0.004730
ZDT2	**0.021928**	0.014674	0.029762	0.006576
ZDT3	**0.013990**	0.005108	0.023994	0.004774
ZDT4	**0.455495**	0.416654	3.098866	2.822281
ZDT6	0.171233	0.117406	**0.106812**	0.055624
DTLZ1	17.965977	7.564753	**16.132116**	6.874782
DTLZ2	0.055607	0.005740	**0.055528**	0.004754

Table 3. Results of $\mathcal{CI}$ for the NSS-GA and NSGA-II

Problem	NSS-GA		NSGA-II	
	average	σ	*average*	σ
ZDT1	**1.000000**	0.000000	0.000000	0.000000
ZDT2	**0.971111**	0.155571	0.004444	0.023934
ZDT3	**0.969534**	0.064908	0.009305	0.022992
ZDT4	**0.686486**	0.322281	0.332336	0.388879
ZDT6	**0.769754**	0.273523	0.144094	0.230425
DTLZ1	**0.590605**	0.147409	0.222936	0.112005
DTLZ2	**0.150000**	0.072019	0.025667	0.022462

were omitted due to space restrictions). With respect to the $\mathcal{S}$ performance measure, our approach obtained better results in 4 test problems, and the NSGA-II obtained better results in the other three. However, since convergence is more important than even distribution of nondominated solutions, we do not consider this to be a major drawback of our proposed approach.

5 Conclusions and Future Work

In this paper, we have introduced a hybridization scheme in which a MOEA (the NSGA-II) is coupled to two direct search methods (Nelder and Mead's method and the golden section method). Our proposed approach (called NSS-GA), was found to be competitive with respect to the original NSGA-II over a set of test functions taken from the specialized literature, when performing only 4,000 fitness function evaluations. As part of our future work, we are interested in experimenting with other direct search methods, such as the Hooke-Jeeves pattern search method and Powell's conjugate direction method. We are also interested in devising mechanisms that help us to decide whether the local search needs to be triggered or not.

Acknowledgments. The first author acknowledges support from CINVESTAV-IPN and CONACyT. The second author acknowledges support from CONACyT through project number 45683-Y.

References

1. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
2. Shukla, P.K.: On Gradient Based Local Search Methods in Unconstrained Evolutionary Multi-objective Optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 96–110. Springer, Heidelberg (2007)

3. Mack, Y., Goel, T., Shyy, W., Haftka, R.: Surrogate Model-Based Optimization Framework: A Case Study in Aerospace Design. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments, pp. 323–342. Springer, Heidelberg (2007)
4. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions Evolutionary Computation 6(2), 182–197 (2002)
5. Spendley, W., Hext, G.R., Himsworth, F.R.: Sequential Application of Simplex Designs in Optimization and Evolutionary Operation. Technometrics 4(4), 441–461 (1962)
6. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. The Computer Journal 7, 308–313 (1965)
7. Ravindran, A., Ragsdell, K., Reklaitis, G.: Engineering Optimization. John Wiley & Sons, Inc., Hoboken (2006)
8. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik 2, 84–90 (1960)
9. van der Corput, J.G.: Verteilungsfunktionen. Akademie van Wetenschappen 38, 813–821 (1935)
10. Hammersley, J.M.: Monte-Carlo methods for solving multivariable problems. Annals of the New York Academy of Science 86, 844–874 (1960)
11. McKinnon, K.I.M.: Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point. SIAM Journal on Optimization 9(1), 148–158 (1998)
12. Trabia, M.B., Lu, X.B.: A Fuzzy Adaptive Simplex Search Optimization Algorithm. Journal of Mechanical Design 123, 216–225 (2001)
13. Rahman, M.K.: An intelligent moving object optimization algorithm for design problems with mixed variables, mixed constraints and multiple objectives. Structural and Multidisciplinary Optimization 32(1), 40–58 (2006)
14. Luersen, M.A., Le Riche, R.: Globalized Nelder-Mead method for engineering optimization. Computers & Structures 82, 2251–2260 (2004)
15. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder–Mead simplex method in low dimensions. SIAM Journal of Optimization 9, 112–147 (1998)
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
17. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, Heidelberg (2005)
18. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)
19. Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics. Massachusetts Institute of Technology, Cambridge, Massachusetts (1995)

SPAM: Set Preference Algorithm for Multiobjective Optimization

Eckart Zitzler, Lothar Thiele, and Johannes Bader

Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
{zitzler,thiele,bader}@tik.ee.ethz.ch

Abstract. This paper pursues the idea of a general multiobjective optimizer that can be flexibly adapted to arbitrary user preferences—assuming that the goal is to approximate the Pareto-optimal set. It proposes the Set Preference Algorithm for Multiobjective Optimization (SPAM) the working principle of which is based on two observations: (i) current multiobjective evolutionary algorithms (MOEAs) can be regarded as hill climbers on set problems and (ii) specific user preferences are often (implicitly) expressed in terms of a binary relation on Pareto set approximations. SPAM realizes a $(1 + 1)$-strategy on the space of Pareto set approximations and can be used with any type of set preference relations, i.e., binary relations that define a total preorder on Pareto set approximations. The experimental results demonstrate for a range of set preference relations that SPAM provides full flexibility with respect to user preferences and is effective in optimizing according to the specified preferences. It thereby offers a new perspective on preference-guided multiobjective search.

1 Motivation

By far most publications within the field of evolutionary multiobjective optimization (EMO) focus on the issue of generating a suitable approximation of the Pareto-optimal set, or Pareto set approximation for short. For instance, the first book on EMO by Kalyanmoy Deb [1] is mainly devoted to techniques of finding multiple trade-off solutions using evolutionary algorithms.

Taking this view, one can state that EMO in general deals with set problems: the search space Ψ consists of all potential Pareto set approximations rather than single solutions, i.e., Ψ is a set of sets. When applying an evolutionary algorithm to the problem of approximating the Pareto-optimal set, the population itself can be regarded as the current Pareto set approximation. The subsequent application of mating selection, variation, and environmental selection heuristically produces a new Pareto set approximation that—in the ideal case—is better than the previous one. In the light of the underlying set problem, the population represents a single element of the search space which is in each iteration replaced by another element of the search space. Consequently, selection and variation can be regarded as a mutation operator on populations resp. sets. Somewhat simplified, one may say that a classical multiobjective evolutionary algorithm

(MOEA) used to approximate the Pareto-optimal set is a $(1,1)$-strategy on a set problem. Furthermore, MOEAs are usually not preference-free. The main advantage of generating methods such as MOEAs is that the objectives do not need to be aggregated or ranked a priori; but nevertheless preference information is required to guide the search, although it is usually weaker and less stringent. In the environmental selection step, for instance, an MOEA has to choose a subset of individuals from the parents and the offspring which constitutes the next Pareto set approximation. To this end, the algorithm needs to know the criteria according to which the subset should be selected, in particular when all parents and children are incomparable, i.e., mutually nondominating. That means the generation of a new population usually relies on set preference information.

These observations led to the concept presented in this paper which separates preference information and search method. Firstly, we regard preference information as an appropriate order on Ψ required to fully specifiy the set problem—this order will here be denoted as set preference relation. A set preference relation provides the information on the basis of which the search is carried out; for any two Pareto set approximations, it says whether one set is better or not. Secondly, we propose a general, extended $(1+1)$-strategy for this set problem which is only based on pairwise comparisons of sets in order to guide the search. The resulting algorithm (SPAM) is fully independent of the set preference relation used and thereby decoupled from the user preferences.

This complete separation of concerns is the novelty of the suggested approach. It builds upon the idea presented in [2], but is is more general—as it is not restricted to a single binary quality indicator—and possess in addition desirable convergence properties. Furthermore, there are various studies that focus on the issue of preference articulation in EMO, in particular integrating additional preferences such as priorities, goals, and reference points [3,4,5,6,7,8,9]. However, these studies mainly cover preferences on solutions and not preferences on sets, and the search procedures used are based on hard-coded preferences.

In the following, we first discuss the issue of designing set preference relations and then present the full SPAM method. Finally, simulation results are provided and compared for several example set preference relations.

2 Set Preference Relations

Consider a multiobjective optimization problem with the decision space X, the objective space Z, n objectives $f_1, \ldots, f_n$ to be minimized, and a relation $\leq$ on Z, which induces a preference relation $\preceq$ on X with $a \preceq b :\Leftrightarrow f(a) \leq f(b)$ for $a, b \in X$. This problem is transformed into a corresponding set problem where the search space Ψ includes all possible solution sets $A \subseteq X$, i.e., $\Psi = 2^X$. The preference relation $\preceq$ can be used to define a corresponding set preference relation $\preccurlyeq$ on Ψ where

$$A \preccurlyeq B :\Leftrightarrow \forall b \in B \, \exists a \in A : \; a \preceq b$$

for all Pareto set approximations $A, B \in \Psi$. Here, we will assume that weak Pareto dominance, represented by $\preceq_{\text{par}}$ and $\preccurlyeq_{\text{par}}$, is the underlying preference relation resp. set preference relation. Most existing MOEAs are designed for such a type of set problem where the goal is to find a good Pareto set approximation $A \in \Psi$.

The set preference relation deduced from the preference relation on solutions is usually not total, i.e., there are incomparable Pareto set approximations which are hard to deal with by any optimization method. Therefore, additional preferences are needed to refine $\preccurlyeq$ such that no incomparable pairs remain. Next, we will discuss principles to design set preference relation that represent total preorders and then provide several example relations.

2.1 Refinements and Sequences

Unary quality indicators are a possible means to construct set preference relations that are total preorders. They represent set quality measures that map each set $A \in \Psi$ to a real number $I(A) \in \mathbb{R}$. Given an indicator I, one can define the corresponding set preference relation as $A \preccurlyeq_{\mathrm{I}} B := (I(A) \leq I(B))$ where we assume that smaller indicator values stand for higher quality, in other words, A is as least as good as B if the indicator value of A is not larger than the one of B. For instance, several recent approaches make implicitly use of the unary hypervolume indicator in this way [10,11,12]. Alternatively, one may consider binary quality indicators that assign a real value to ordered pairs of sets (A, B) with $A, B \in \Psi$. Assuming that smaller indicator values stand for higher quality, a corresponding set preference relation can be defined as $A \preccurlyeq_{\mathrm{I}} B := (I(A, B) \leq I(B, A))$. For instance, IBEA [2] uses this type of preference information.

When defining set preference relations based on indicators (or using other principles), we would like to guarantee that weak Pareto dominance is not violated, i.e., $\preccurlyeq_{\mathrm{I}}$ should refine $\preccurlyeq_{\mathrm{par}}$. This can be formalized as follows.

Definition 2.1. *Given a set Ψ. Then the preference relation $\preccurlyeq_{ref}$ refines $\preccurlyeq$ if for all $A, B \in \Psi$ we have*

$$(A \preccurlyeq B) \wedge (B \not\preccurlyeq A) \Rightarrow (A \preccurlyeq_{ref} B) \wedge (B \not\preccurlyeq_{ref} A)$$

That means a set that is strictly better than another set in the original set preference relation should remain strictly better in the refined relation. The hypervolume indicator [13,12], for instance, induces a refinement of weak Pareto dominance, cf. [14,15]. Many other indicators only fulfill a weaker property which we here denote as weak refinement.

Definition 2.2. *Given a set Ψ. Then the set preference relation $\preccurlyeq_{ref}$ weakly refines $\preccurlyeq$ if for all $A, B \in \Psi$ we have*

$$(A \preccurlyeq B) \wedge (B \not\preccurlyeq A) \Rightarrow (A \preccurlyeq_{ref} B)$$

A weak refinement may make two sets A and B indifferent ($A \preccurlyeq_{\mathrm{ref}} B \ \wedge \ B \preccurlyeq_{\mathrm{ref}} A$), although A is actually strictly better than B; this is for instance the case for the unary epsilon indicator [15]. Nevertheless, a weak refinement never contradicts the original order, i.e., B cannot be strictly preferable to A with regard to

$\preccurlyeq_{\text{ref}}$, whenever A is strictly better than B regarding $\preccurlyeq$. However, many practically interesting indicators do not induce a weak refinement of the weak Pareto dominance relation.

For optimization purposes, it is desirable to have a set preference relation that represents a refinement of the dominance relation because this is a prerequisite to achieve convergence to the Pareto-optimal set, see [16]. The following construction shows how such refinements can be defined on the basis of arbitrary indicators; it resembles the concept of hierarchy used in [3] for pairs of solutions, but here (a) we are dealing with preference relations on sets and (b) the hierarchical construction is different.

Definition 2.3. *Given a set Ψ and a sequence S of k preference relations over Ψ with $S = (\preccurlyeq^1, \preccurlyeq^2, \ldots, \preccurlyeq^k)$. Then the preference relation $\preccurlyeq_S$ associated with S is defined as follows: Let $A, B \in \Psi$. Then $A \preccurlyeq_S B$ if and only if $\exists 1 \leq i \leq k$ such that the following two conditions are satisfied:*

(i) $(i < k \ \wedge \ (A \preccurlyeq^i B \wedge B \not\preccurlyeq^i A)) \vee (i = k \ \wedge \ (A \preccurlyeq^k B))$
(ii) $\forall 1 \leq j < i : (A \preccurlyeq^j B \ \wedge \ B \preccurlyeq^j A)$

With this definition, we can derive the following procedure to determine $A \preccurlyeq_{\text{S}} B$ for two sets A and B:

– Start from the first preference relation, i.e. $j = 1$. Repeat the following step: If A and B are indifferent with respect to $\preccurlyeq^j$, then increase j to point to the next relation in the sequence if it exists.
– If the final j points to the last preference relation ($j = k$), then set $A \preccurlyeq_{\text{S}} B \Leftrightarrow A \preccurlyeq^k B$. Otherwise, set $A \preccurlyeq_{\text{S}} B \Leftrightarrow A \prec^k B$.

This procedure allows to use indicators inducing only weak refinements or no refinements at all in combination with refinements; the resulting set preference relation is again a refinement.

Theorem 2.4. *Given a sequence of preference relations according to Def. 2.3. Suppose there is a $k' \leq k$ such that $\preccurlyeq^{k'}$ is a refinement of a given preference relation $\preccurlyeq$. In addition, all relations $\preccurlyeq^j$, $1 \leq j < k'$ are weak refinements of $\preccurlyeq$ and all relations $\preccurlyeq^j$, $k' < j \leq k$ are preorders. Then $\preccurlyeq_S$ is a refinement of $\preccurlyeq$.*

For reasons of space limitations, the proof for this theorem is omitted here; it is provided in [16]. Fig. 1 visualizes the resulting construction principle. This will be used in Section 2.2 to design indicators combinations representing different types of preference information.

2.2 Design of Indicator-Based Relations

In the following, we present some examples for combined set preference relations that illustrate different application scenarios. All of these relations are refinements of the set preference relation $\preccurlyeq_{\text{par}}$.

The first combination is based on the unary epsilon indicator $I_{\epsilon 1}$ [15] with a reference set R in objective space which is defined as $I_{\epsilon 1}(A) = E(A, R)$ with

$$S = (\underbrace{\preccurlyeq^1, \dots, \preccurlyeq^{k'-1}}_{\text{weak refinement}}, \overset{\text{refinement}}{\preccurlyeq^{k'}}, \underbrace{\preccurlyeq^{k'+1}, \dots, \preccurlyeq^{k}}_{\text{preorder}})$$

Fig. 1. Representation of the hierarchical construction of refinements according to Theorem 2.4

$E(A, R) = \max_{r \in R} \ \min_{a \in A} \ \epsilon(a, r)$ where $\epsilon(a, r) = \max\{f_i(a) - r_i \mid 1 \leq i \leq n\}$ and r_i is the ith component of the objective vector r. Since this indicator induces only a weak refinement of the weak Pareto-dominance relation $\preccurlyeq_{\text{par}}$, we will use the hypervolume indicator to distinguish between sets indifferent with respect $I_{\epsilon 1}$. The resulting set preference relation is denoted as $\preccurlyeq_{\epsilon 1, H}$.

The second combination uses the R_2 indicator proposed in [17] for which the following definition is used here:

$$I_{R2}(A) = R_2(A, R) = \frac{\sum_{\lambda \in \Lambda} u^*(\lambda, R) - u^*(\lambda, f(A))}{|\Lambda|}$$

where the function u^* is a utility function based on the weighted Tchebycheff function $u^*(\lambda, T) = -\min_{z \in T} \max_{1 \leq j \leq n} \lambda_j |z_j^* - z_j|$ and Λ is a set of weight vectors $\lambda \in \mathbb{R}^n$, $R \subset \mathcal{Z}$ is a reference set, and $z^* \in \mathcal{Z}$ is a reference point. In this paper, we will set $R = \{z^*\}$. Also the R_2 indicator provides only a weak refinement; as before, the hypervolume indicator is added in order to achieve a refinement. This set preference relation will be denoted as $\preccurlyeq_{R2,H}$.

Third combination: The previous two indicator combinations couple a weak refinement with a refinement. To demonstrate that also non-refining indicators can be used, we propose the following sequence of indicators $S = (I_H, I_C, I_D)$ where I_C measures the largest distance of a solution to the closest minimal element in a set and I_D reflects the diversity of the solutions in the objective space. The latter two indicators, which both do not induce weak refinements of $\preccurlyeq_{\text{par}}$, are defined as follows: $I_C(A) = \max_{a \in A} \min_{b \in \text{Min}(A, \preceq)} dist(f(a), f(b))$ and

$$I_D(A) = \max_{a \in A} \left(\frac{1}{nn_1(a, A \setminus \{a\})} + \frac{1}{nn_2(a, A \setminus \{a\})} \right)$$

where $nn_1(a, B) = \min_{b \in B} dist(f(a), f(b))$ gives the smallest and $nn_2(a, B) = \max_{c \in B} \min_{b \in B \setminus \{c\}} dist(f(a), f(b))$ the second smallest distance of a to any solution in B. For the distance function $dist(z^1, z^2)$, Euclidean distance is used here, i.e., $dist(z^1, z^2) = \sqrt{\sum_{1 \leq i \leq n} (z_i^1 - z_i^2)^2}$. The I_C indicator resembles the generational distance measure proposed in [18] and I_D resembles the nearest neighbor niching mechanism in SPEA2 [19]. We will refer to the overall set preference relation as $\preccurlyeq_{H,C,D}$. According to Theorem 2.4, $\preccurlyeq_{H,C,D}$ is a refinement of $\preccurlyeq_{\text{par}}$.

Finally, note that set preference relations may be insensitive to dominated solutions in a set, i.e., adding dominated solutions to A or B does not affect the relation between these two sets. This holds for instance for the set preference relations induced by the hypervolume indicator and other popular quality indicators. Nevertheless, to guide the search efficiently it is crucial that prefered solutions are taken into account. One possibiliy is to integrate indicators that are sensitive to dominated solutions such as I_C and I_D defined above. Alternatively, the sets can be partitioned into dominance classes to which the set preference relation is applied subsequently. More precisely, we here use nondominated sorting [20,21] for partitioning and then use the same construction as in Theorem 2.4: to compare A and B with respect to $\preccurlyeq$ we first compare only the nondominted fronts; if this comparison yields indifference, then the second level of nondominance is considered to decide whether A or B is better, and so forth. Whenever this principle of partitioning is used, we write $\preccurlyeq^{\mathsf{minpart}}$; note that $\preccurlyeq^{\mathsf{minpart}}$ is a refinement of $\preccurlyeq$.

3 A General Set Preference Guided Search Algorithm

In the following, we introduce the **S**et **P**reference **A**lgorithm for **M**ultiobjective Optimization (SPAM) which can be used with any set preference relation and resembles a standard hill climber with the difference that two new elements of the search space Ψ are created using two types of mutation operators. The main part of SPAM is given by Algorithm 1.

Starting with a randomly chosen set $P \in \Psi_m$ of size m, first a random mutation operator is applied to generate another set P'. This operator should be designed such that every element in Ψ could be possibly generated, i.e., the neighborhood is in principle the entire search space. In practice, the operator will usually have little effect on the optimization process; however, its property of exhaustivness is important from a theoretical perspective, in particular to show convergence, see [16].

Second, a heuristic mutation operator is employed. This operator mimics the mating selection, variation, and environmental selection steps as used in most MOEAs. The goal of this operator is to create a third set $P'' \in \Psi$ that is better than P in the context of a predefined set preference relation $\preccurlyeq$. However, since it is heuristic it cannot guarantee to improve P; there may be situations where it

Algorithm 1. SPAM Main Loop

```
Require: set preference relation $\preccurlyeq$
1: generate initial set $P$ of size $m$, i.e., randomly choose $A \in \Psi_m$ and set $P \leftarrow A$
2: while termination criterion not fulfilled do
3:     $P' \leftarrow \mathit{randomSetMutation}(P)$
4:     $P'' \leftarrow \mathit{heuristicSetMutation}(P)$
5:     if $P'' \preccurlyeq P$ then
6:         $P \leftarrow P''$
7:     else if $P' \preccurlyeq P$ then
8:         $P \leftarrow P'$
9: return $P$
```

is not able to escape local optima of the landscape of the underlying set problem. Finally, P is replaced by P'', if the latter is weakly preferable to the former; otherwise, P is either replaced by P' (if $P' \preccurlyeq P$) or remains unchanged. Note that in the last step, weak preferability ($\preccurlyeq$) and not preferability ($\prec$) needs to be considered in order to allow the algorithm to cross landscape plateaus, cf. [22].

For the mutation operators, we propose Algorithms 2 and 3. Algorithm 2 (random set mutation) randomly chooses k decision vectors from $\mathcal{X}$ and uses them to replace k elements in P.[1] Algorithm 3 (heuristic set mutation) generalizes the iterative truncation procedures used in NSGA-II [23], SPEA2 [19], and others. First, k new solutions are created based on P; this corresponds to mating selection plus variation in a standard MOEA. While the variation is problem-specific, for mating selection either uniform random selection (used in the following) or fitness-based selection can be used (using the fitness values computed by Algorithm 4). Then, these k solutions are added to P, and finally the resulting set of size $m + k$ is iteratively truncated to size m by removing the solution with the worst fitness values in each step. Here, the fitness value of $a \in P$ reflects the loss in quality for the entire set P if a is deleted: the lower the fitness, the larger the loss.

Algorithm 2. Random Set Mutation

```
1: procedure randomSetMutation($P$)
2:     randomly choose $r_1, \ldots, r_k \in \mathcal{X}$ with $r_i \neq r_j$
3:     randomly select $p_1, \ldots, p_k$ from $P$ with $p_i \neq p_j$
4:     $P' \leftarrow P \setminus \{p_1, \ldots, p_k\} \cup \{r_1, \ldots, r_k\}$
5:     return $P'$
```

Algorithm 3. Heuristic Set Mutation

```
1: procedure heuristicSetMutation($P$)
2:     generate $r_1, \ldots, r_k \in \mathcal{X}$ based on $P$
3:     $P'' \leftarrow P \cup \{r_1, \ldots, r_k\}$
4:     while $|P''| > m$ do
5:         for all $a \in P''$ do
6:             $\delta_a \leftarrow$ fitnessAssignment(a, P”)
7:         choose $p \in P''$ with $\delta_p = \min_{a \in P''} \delta_a$
8:         $P'' \leftarrow P'' \setminus \{p\}$
9:     return $P''$
```

To estimate how useful a particular solution $a \in P$ is, Algorithm 4 compares all sets $A_i \subset P$ with $|A_i| = |P| - 1$ to $P \setminus \{a\}$ using the predefined set preference relation $\preccurlyeq$. The fewer sets A_i are weakly preferable to $P \setminus \{a\}$, the better the set $P \setminus \{a\}$ and the less important is a. This procedure has a runtime complexity of $\mathcal{O}((m + k)g)$, where g stands for the runtime needed to compute the preference

[1] Note that for both mutation operators the same k is used here, although they can be chosen independently. The safe version ($k = m$) for the random mutation operator means that a random walk is carried out on Ψ.

Algorithm 4. Fitness Assignment

```
1: procedure fitnessAssignment($a, P''$)
2:     $\delta_a \leftarrow 0$
3:     for all $b \in P''$ do
4:         if $P'' \setminus \{b\} \preccurlyeq P'' \setminus \{a\}$ then
5:             $\delta_a \leftarrow \delta_a + 1$
6:     return $\delta_a$
```

relation comparisons which usually depends on $m+k$ and the number of objective functions. It can be made faster, when using unary indicators, see [16].

4 Experiments

This section investigates the practicability of the proposed approach. The main questions are: (i) can different user preferences be expressed in terms of set preference relations, (ii) is it feasible to use a general search algorithm for arbitrary set preference relations, i.e., is SPAM effective in finding appropriate sets, and (iii) how well are set preference relations suited to guide the optimization process? However, the purpose is not to carry out a performance comparison of SPAM to existing MOEAs; the separation of user preferences and search algorithm is the focus of our study.

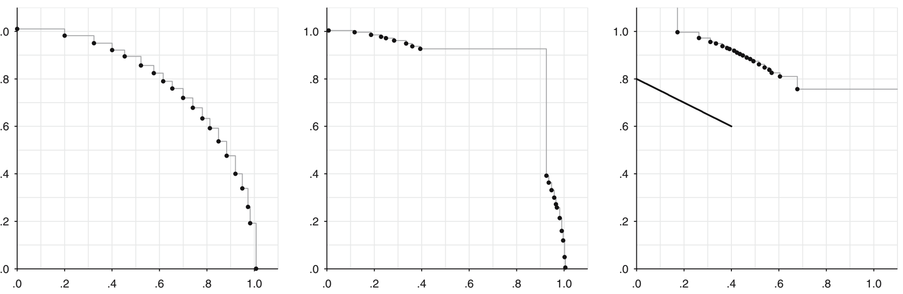
In the following, we consider the different set preference relations presented in Section 2.2 for integration in SPAM where $\preccurlyeq^{\mathsf{minpart}}_{R2,H}$ is parameterized to focus on the outer regions of the Pareto front.[2] In addition, the relations $\preccurlyeq^{\mathsf{minpart}}_{H}$ and $\preccurlyeq^{\mathsf{minpart}}_{\epsilon}$ induced by the unary hypervolume indicator resp. the binary epsilon indicator are used. All of them are refinements of the set dominance relation $\preccurlyeq_{\mathrm{par}}$. As reference algorithms, NSGA-II [23] and IBEA[3] [2] are used. The test problem is DTLZ2 [24] with 20 decision variables and 2 resp. 5 objectives [4]. To compare the outcomes of the algorithms with respect to multiple runs (in this study 30 runs) statistically, we use the Mann-Whitney U test where the significance of the test statistics U is calculated on the basis of the one-tailed normal approximation, correcting the variance for ties, see [16] for details. Furthermore, multiple testing issues need to be taken into account when comparing multiple algorithms with each other; here, the significance levels are Bonferroni corrected.

Figure 2 shows the Pareto-set approximations generated by SPAM with three selected set preference relations. The plots well reflect the chosen user preferences: (a) a set maximizing hypervolume, (b) focus on the extremes using corresponding weight combinations, and (c) closeness to a given reference set. This

[2] The full parameterization of the indicators is given in [16].

[3] With parameters $\kappa = 0.05$ and $\rho = 1.1$.

[4] Other parameters: set size / population size $m = 20$ (visual comparisons) resp. $m = 50$ (statistical comparisons); newly created solutions / offspring individuals $k = 20$ resp. $k = 50$; number of iterations 1000; further details are provided in [16].



(a) SPAM with $\preccurlyeq_{H}^{\mathsf{minpart}}$ (b) SPAM with $\preccurlyeq_{R2,H}^{\mathsf{minpart}}$ (c) SPAM with $\preccurlyeq_{\epsilon 1,H}^{\mathsf{minpart}}$

Fig. 2. Pareto-set approximations found after 1000 generations on a biobjective DTLZ2 problem for a set size / population size of $m = 20$. All algorithms were started with the same initial set / population.

demonstrates that SPAM is in principle capable of optimizing towards the user preferences that are encoded in the corresponding set preference relation.

The quantitative comparisons for all set preference relations are provided in Table 1. The hypothesis is that SPAM used in combination with a specific set preference relation $\preccurlyeq_A$ (let us say SPAM-A) yields better Pareto set approximations than if used with any other set preference relation $\preccurlyeq_B$ (let us say SPAM-B)—better here means with respect to $\preccurlyeq_A$. Ideally, for every set A generated by SPAM-A and every set B generated by SPAM-B, it would hold $A \preccurlyeq_A B$ or even $A \prec_A B$. Clearly, this describes an ideal situtation. A set preference relation that is well suited for representing certain preferences may not be well suited for search per se. With only few exceptions, the above hypothesis is confirmed: using $\preccurlyeq_A$ in SPAM yields the best Pareto-set approximations with regard to $\preccurlyeq_A$, independently of the problem and the number of objectives under consideration. These results are highly significant at a significance level of 0.001.

Concerning the exceptions, first it can be noticed that there is no significant difference between $\preccurlyeq_{H}^{\mathsf{minpart}}$ and $\preccurlyeq_{H,C,D}$ when used in SPAM—both times, the hypervolume indicator value is optimized. This actually shows that dominance class partitioning may be replaced by a corresponding sequence of quality indicators. Second, the algorithm based on the set preference relation $\preccurlyeq_\epsilon$ using the binary epsilon indicator performs slighlty worse than IBEA with respect to $\preccurlyeq_\epsilon$. This is not suprising since IBEA has been designed mainly for the epsilon indicator and exploits certain characteristics; for instance, all population members are compared to each other and not only those in the current front. In addition, SPAM with the binary epsilon indicator performs significantly worse than SPAM with any of the two hypervolume-based relations $\preccurlyeq_{H}^{\mathsf{minpart}}$ and $\preccurlyeq_{H,C,D}$ in the case of two objectives. This may indicate that the binary epsilon indicator is not sensitive enough to differentiate between small improvements. That means that in Step 7 of Algorithm 3 too many solutions may achieve the minimum δ-value, and therefore a choice needs to be done at random.

Table 1. Pairwise statistical comparison of 30 runs per algorithm after 1000 generations. In the notation $U : U'$, U (resp. U') stands for the number of times a set generated by algorithm A (resp. B) beats a set of algorithm B (resp. A) with regard to the test relation associated with the corresponding row. A star next to these numbers indicates a significant difference, the few cases where this was not the case are shown in bold. Per cell, the upper number pair corresponds to the biobjective DTLZ2 problem, the lower number pairs to the 5-objective DTLZ2 problem.

alg. A \ alg B.		SPAM with set preference relation ...					IBEA	NSGA-II	test relation
		$\preccurlyeq_{H}^{\mathsf{minpart}}$	$\preccurlyeq_{R2,H}^{\mathsf{minpart}}$	$\preccurlyeq_{\epsilon 1,H}^{\mathsf{minpart}}$	$\preccurlyeq_{\epsilon}^{\mathsf{minpart}}$	$\preccurlyeq_{H,C,D}$			
SPAM with set preference relation ...	$\preccurlyeq_{H}^{\mathsf{minpart}}$	-	900: 0*	900: 0*	900: 0*	**456:444**	900: 0*	900: 0*	$\preccurlyeq_{H}$
		-	900: 0*	900: 0*	900: 0*	**445:455**	900: 0*	900: 0*	
	$\preccurlyeq_{R2,H}^{\mathsf{minpart}}$	900: 0*	-	900: 0*	900: 0*	900: 0*	900: 0*	900: 0*	$\preccurlyeq_{R2,H}$
		900: 0*	-	900: 0*	900: 0*	900: 0*	900: 0*	900: 0*	
	$\preccurlyeq_{\epsilon 1,H}^{\mathsf{minpart}}$	900: 0*	900: 0*	-	900: 0*	889: 1*	900: 0*	900: 0*	$\preccurlyeq_{\epsilon 1,H}$
		891: 9*	900: 0*	-	900: 0*	897: 3*	900: 0*	900: 0*	
	$\preccurlyeq_{\epsilon}^{\mathsf{minpart}}$	**63:837**	900: 0*	900: 0*	-	**81:819**	**274:626**	896: 4*	$\preccurlyeq_{\epsilon}$
		60:840	900: 0*	900: 0*	-	**57:843**	**349:551**	889: 11*	
	$\preccurlyeq_{H,C,D}$	**444:456**	900: 0*	900: 0*	853: 47*	-	820: 80*	900: 0*	$\preccurlyeq_{H,C,D}$
		455:445	900: 0*	900: 0*	900: 0*	-	900: 0*	900: 0*	

* Preference is significant at the 0.001 level (1-tailed, Bonferroni-adjusted).

5 Conclusions

This paper proposed a general way to separate preference formalization from algorithm design and presented SPAM, a flexible multiobjective optimizer, which is basically a hill climber and generalizes the concepts found in most modern MOEAs. SPAM can be used in combination with any type of set preference relation and thereby offers full flexibility for the decision maker. Furthermore, a novel scheme to design set preference relations by putting multiple quality indicators in sequence was introduced.

References

1. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester (2001)
2. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN VIII 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
3. Fonseca, C.M., Fleming, P.J.: Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part I: A Unified Formulation. IEEE Transactions on Systems, Man, and Cybernetics 28(1), 26–37 (1998)
4. Branke, J., Kaußler, T., Schmeck, H.: Guidance in Evolutionary Multi-Objective Optimization. Advances in Engineering Software 32, 499–507 (2001)
5. Cvetković, D., Parmee, I.C.: Preferences and their Application in Evolutionary Multiobjective Optimisation. IEEE Transactions on Evolutionary Computation 6(1), 42–57 (2002)
6. Branke, J., Deb, K.: Integrating User Preferences into Evolutionary Multi-Objective Optimization. Technical Report 2004004, Indian Institute of Technology,

Kanpur, India (2004); In: Jin, Y. (ed). Knowledge Incorporation in Evolutionary Computation, pp. 461–477. Springer, Heidelberg (2004)

7. Deb, K., Sundar, J.: Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms. In: Keijzer, M., et al. (eds.) Conference on Genetic and Evolutionary Computation (GECCO 2006), pp. 635–642. ACM Press, New York (2006)
8. Rachmawati, L., Srinivasan, D.: Preference Incorporation in Multi-objective Evolutionary Algorithms: A Survey. In: IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada, pp. 3385–3391. IEEE Press, Los Alamitos (2006)
9. Mehnen, J., Trautmann, H., Tiwari, A.: Introducing User Preference Using Desirability Functions in Multi-Objective Evolutionary Optimisation of Noisy Processes. In: IEEE Congress on Evolutionary Computation (CEC 2007), pp. 2687–2694. IEEE Press, Los Alamitos (2007)
10. Emmerich, M., Beume, N., Naujoks, B.: An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)
11. Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-objective Optimization. Evolutionary Computation 15(1), 1–28 (2007)
12. Zitzler, E., Brockhoff, D., Thiele, L.: The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In: Obayashi, S., et al. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)
13. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
14. Knowles, J., Corne, D.: On Metrics for Comparing Non-Dominated Sets. In: Congress on Evolutionary Computation (CEC 2002), pp. 711–716. IEEE Computer Society Press, Piscataway (2002)
15. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)
16. Zitzler, E., Thiele, L., Bader, J.: On Set-Based Multiobjective Optimization. Technical Report 300, Computer Engineering and Networks Laboratory, ETH Zurich (2008)
17. Hansen, M.P., Jaszkiewicz, A.: Evaluating the quality of approximations of the non-dominated set. Technical report, Institute of Mathematical Modeling, Technical University of Denmark, IMM Technical Report IMM-REP-1998-7 (1998)
18. Veldhuizen, D.A.V., Lamont, G.B.: Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. Evolutionary Computation 8(2), 125–147 (2000)
19. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou, K., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), pp. 95–100 (2002)
20. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
21. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2(3), 221–248 (1994)

22. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do Additional Objectives Make a Problem Harder? In: Thierens, D., et al. (eds.) Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 765–772. ACM Press, New York (2007)
23. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: Schoenauer, M., et al. (eds.) PPSN VI 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
24. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Congress on Evolutionary Computation (CEC 2002), pp. 825–830. IEEE Press, Los Alamitos (2002)

Modeling Human Expertise on a Cheese Ripening Industrial Process Using GP

Olivier Barrière[1], Evelyne Lutton[1],
Cedric Baudrit[2], Mariette Sicard[2], Bruno Pinaud[2], and Nathalie Perrot[2]

[1] INRIA Saclay - Ile-de-France
Parc Orsay Université, 4, rue Jacques Monod
91893 ORSAY Cedex, France
{Olivier.Barriere,Evelyne.Lutton}@inria.fr
http://apis.saclay.inria.fr
[2] UMR782 Génie et Microbiologie des Procédés Alimentaires.
AgroParisTech, INRA, F-78850 Thiverval-Grignon, France
{cbaudrit,mariette.sicard,bruno.pinaud,nperrot}@grignon.inra.fr

Abstract. Industrial agrifood processes often strongly rely on human expertise, expressed as know-how and control procedures based on subjective measurements (color, smell, texture), which are very difficult to capture and model. We deal in this paper with a cheese ripening process (of french Camembert), for which experimental data have been collected within a cheese ripening laboratory chain. A global and a monopopulation cooperative/coevolutive GP scheme (Parisian approach) have been developed in order to simulate phase prediction (i.e. a subjective estimation of human experts) from microbial proportions and Ph measurements. These two GP approaches are compared to Bayesian network modeling and simple multilinear learning algorithms. Preliminary results show the effectiveness and robustness of the Parisian GP approach.

1 Introduction

This study is part of the large INCALIN research project, whose goal is the modeling of agrifood industrial processes[1]. The competitive challenge to which agrifood industries are facing is related to quality and sustainability of food products. The aim of the INCALIN project is to build decision support tools to manage the manufacturing process as a whole. Current knowledge on industrial agrifood processes are focussed on microbiological, mechanistic, sensorial or physicochemical changes, and are expressed in various ways: databases, mathematical models, and know-how of operators-experts in terms of formal or unformal reasoning. Among the fragmented knowledge available, the human-expert knowledge is certainly the most challenging to capture.

We focus in this paper on a cheese ripening process (section 2): The cheese, during ripening, is an ecosystem (a bio-reactor) that is extremely complex to be modeled as a whole, and where human experts operators have a decisive role. The modifications of substrate under the action of several populations of micro-organisms is only

[1] Supported by the French ANR-PNRA fund.

partially known, and various macroscopic models have been experimented to embed expert knowledge, like expert systems [12,13,11], neural networks [14,17], mechanistic models [1,20], or dynamic Bayesian networks [4].

The major problem common to these techniques is related to the sparseness of available data: collecting experimental data is a long and difficult process, and resulting data sets are often uncertain or even erroneous. The precision of the resulting model is often limited by the small number of valid experimental data, and parameteter estimation procedures have to deal with incomplete, sparse and uncertain data. In this context we consider stochastic optimisation techniques, like evolutionary techniques, which have been proven successful on several complex agrifood problems [3,8,21].

The idea developed in this study is based on the following question: is it possible to capture (learn) in a satisfying way an expert knowledge with help of a model evolved by genetic programming, for a complex cheese ripening process ?

The first step in this direction aims at comparing a part of a reference dynamic Bayesian model whose structure is based on expert knowledge (section 2) with evolved GP estimators, using a global strategy (section 3) and a cooperative/coevolutive strategy (Parisian GP, section 4). Experimental results (section 5) prove the efficacy of GP approaches to estimate the phase parameter of the process (currently made "at hand" in industrial chains). Section 6 then sketches the next steps of the study in order to build an efficient model of the whole cheese ripening process.

2 The Camembert-Cheese Ripening Process

For soft-mould cheese the most important biochemical phenomena occur during ripening. Relationships between microbiological and physicochemical changes depend on environmental conditions (*e.g.* temperature, relative humidity ...) [15] and influence the quality of ripened cheeses [9,16]. A ripening expert is able to estimate the current state of some of the complex reactions at a macroscopic level through its perceptions. Control decisions are then generally based on these subjective but robust expert measurements.

Experimental procedures in laboratories ("model cheeses") use pasteurized milk inoculated with *Kluyveromyces marxianus* (*Km*), *Geotrichum candidum* (*Gc*), *Penicillium camemberti* (*Pc*) and *Brevibacterium auriantiacum* (*Ba*) under aseptic conditions (detailed in [16]).

Experts use their senses to follow cheese ripening and they probably aggregate in a complex way these information to regulate the evolution of the process. An important information for parameter regulation is the subjective estimation of the current state of the ripening process, discretised in four phases:

– **Phase 1** is characterized by the surface humidity evolution of cheese (drying process). At the beginning, the surface of cheese is very wet and evolves until it presents a rather dry aspect. The cheese is white with an odor of fresh cheese.
– **Phase 2** begins with the apparition of a *P. camemberti*-coat (*i.e* the white-coat at the surface of cheese), it is characterized by a first change of color and a "mushroom" odor development.

- **Phase 3** is characterized by the thickening of the creamy under-rind. *P. camemberti* cover all the surface of cheeses and the color is light brown.
- **Phase 4** is defined by strong ammoniac odor perception and the dark brown aspect of the rind of cheese.

These four steps are representative of the main evolution of the cheese during ripening. The expert's knowledge is obviously not limited to these four stages. But these stages help to evaluate the whole dynamics of ripening and to detect drift from the standard evolution.

3 Phase Estimation Using GP

The interest of evolutionary optimisation methods for the resolution of complex problems related to agrifood has been proved by various recent publications. For example [3] uses genetic algorithms to identify the smallest discriminant set of variables to be used in certification process for an italian cheese (validation of origin labels). [8] used GP to select the most significant wavenumbers produced by a Fourier transform infrared spectroscopy measurement device, in order to build a rapid detector of bacterial spoilage on beef. And a recent overview on optimisation tools in food industries [21] mentions works based on evolutionary approaches.

In a previous work on cheese ripening modeling [4,19], a dynamic bayesian network has been built, using human expert knowledge, to represent the macroscopic dynamic of each variable. The phase the network is in at time t plays a determinant role for the prediction of the variables at time $t + 1$. Moreover, four relevant variables have been identified, the derivative of pH, la, *Km* and *Ba* at time t, allowing to predict phase at time $t + 1$.

In this paper, we will focus on the phase estimation process: a genetic programming approach is used to search for a convenient formula that links the four derivatives of micro-organisms proportions to the phase at each time step t (static model), without *a priori* knowledge of the phase at $t-1$. This problem is a symbolic regression one, however, it has to be noted that the small number of samples and their irregular distribution make it difficult. Results will be compared with the performances of a static Bayesian network, extracted from the DBN of [4], and with a very simple learning algorithms (multilinear prediction, see section 5).

3.1 Search Space

The derivatives of four variables will be considered, i.e. the derivative of pH (acidity), la (lactose proportion), *Km* and *Ba* (lactic acid bacteria proportions, see section 2), for the estimation of the phase (static problem). The GP will search for a phase estimator $\widehat{Phase}(t)$, i.e. a function defined as follows:

$$\widehat{Phase}(t) = f(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t})$$

The function set is made of arithmetic operators: $\{+, -, *, /, \hat{}, log\}$, with protected $/$ and log, and logical operators $\{if, >, <, =, and, or, xor, not\}$ in order to allow complex estimation formula.

The terminal set is made of the four partial derivatives plus real constants. The constant's values are not limited, but randomly initialised using one of the following laws $\mathcal{U}\,[0,1]$, $-\mathcal{U}\,[0,1]$, $\mathcal{N}\,(0,1)$ ($\mathcal{U}$ is the uniform law, and $\mathcal{N}$ the normal law).

3.2 Fitness Function

Available data are shared in two sets: learning set and validation set, that are randomly chosen within the available data set for each run. The 16 available experiences are thus randomly shared between learning and validation sets. The size of the learning set vary from 10 to 15 experiments, while the size of the corresponding validation set vary from 6 to 1 experiment (see section 5).

The fitness function, *to be minimised*, is made of a factor that measures the quality of the fitting on the learning set, plus a "parsimony" penalisation factor in order to minimize the size (the number of nodes, actually) of the evolved structures (to avoid bloat). It is divided by the number of variables involved in the evaluated tree in order to favour structures that embed all four variables of the problem (this is a requirement of biologists ; experiments also show that recognition results are better with this constraint):

$$fitness = \frac{\sum_{learning\_set} \left| f(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t}) - Phase(t) \right| + W\#Nodes}{\#Variables + 1}$$

The parameter W has been experimentally tuned, the optimal value ($W = 1$) favours evolution of structures with 30 to 40 nodes.

3.3 Genetic Operators

A classical tree crossover (exchange of subtrees from a randomly chosen node) has been used with probability p_c (defined per tree), as a means of evolving the structure of the tree. Two types of mutations have been used:

- **A subtree mutation** (mutation of the structure), that randomly rebuilt a new subtree from a randomly chosen node, applied with probability p_{sm} (defined per tree),
- **A point mutation** (mutation of nodes content), applied with probability p_{cm} (also defined per tree) that does not modify the structure, but randomly changes the content of each node of the tree within the set of compatible functions or terminals (arity constraints). The probabilities (defined per node) are detailed in table 1. Real values are considered separately and undergo a real mutation with probability p_{rm} as a multiplicative perturbation according to a χ^2 law of parameter N:

$$x' = x \frac{\sum_{i=1}^{N} \mathcal{N}(0,1)^2}{N}$$

p_{rm} and N vary linearly according to generations, from 0.1 to 0.5 for p_{rm}, and from 1 to 1000 for N, in order to start with rather unfrequent large radius mutations and finish with more frequent mutations with smaller radius.

Table 1. Probabilities of point mutation operators

From	to	probability
operator	operator	0.1
variable	variable	0.1
variable	constant	0.05
constant	variable	0.05
constant	constant	p_{rm}: 0.1 to 0.5
		N: 1 to 1000

Crossover, subtree and point mutation probabilities vary along evolution according to the adapting scheme[6] available in the GPLAB toolbox[10]. p_c, p_{sm} and pcm are initially fixed to $\frac{1}{3}$, and are updated according statistics of success of the various operators computed on a tuneable window of past generations.

4 Phase Estimation Using a Parisian GP

Cooperative co-evolution techniques rely on the imitation of cooperative capabilities of natural populations, and their ability to build solutions via a cooperation process. These techniques are starting to be used with success on learning problems, see [2] for a recent reference on the topic. The large majority of these approaches deals with a coevolution process that happens between a fixed number of separated populations. We experiment here a different implementation of cooperative coevolution principles, known as the Parisian approach [5,18], that uses cooperation mechanisms within a *single* population. It is based on a two-level representation of an optimization problem, in the sense that an individual of a Parisian population represents only a part of the problem solution. An aggregation of multiple individuals must be built in order to obtain a solution at hand. In this way, the co-evolution of the whole population (or a major part of it) is favoured instead of the emergence of a single best individual, as in classical evolutionary schemes. The motivation is to make a more efficient use of the genetic search process, and reduce the computational expense. Successful applications of such a scheme usually rely on a lower cost evaluation of the partial solutions (i.e. the individuals of the population), while computing the full evaluation only once at each generation.

Phase estimation can actually be split into 4 combined (and simpler) phase detection trees. The structures searched are then binary output functions (or binarised functions) that are able to characterize one of the four phases. A global solution being made of at least one individual of each phase.

4.1 Search Space and Local Fitness Measurements

We now search for formulas of type: $I(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t})$ with real outputs mapped to binary outputs, via a sign filtering: $(I() > 0) \rightarrow 1$ and $(I() \leq 0) \rightarrow 0$. The functions (except logical ones) and terminal sets, as well as the genetic operators, are the same as in the global approach above.

Using the available samples of the learning set, four values can be computed, in order to measure the capability of an individual I to characterize each phase:

$$k \in \{1,2,3,4\} \quad F_k(I) = 3 \sum_{i,phase=k} \frac{I(sample(i))}{\#Samples_{phase=k}} - \sum_{i,phase\neq k} \frac{I(sample(i))}{\#Samples_{phase\neq k}}$$

i.e. if I is good for representing phase k, then $F_k(I) > 0$ and $F_{\neq k} < 0$

The local fitness value, *to be maximised*, is a combination of three factors:

$$LocalFit = \max\{F_1, F_2, F_3, F_4\} \times \frac{\#Ind}{\#IndPhaseMax} \times \frac{NbMaxNodes}{NbNodes}\bigg|_{\text{if } NbNodes > NbMaxNodes}$$

The first factor is aimed at characterising if individual I is able to distinguish one of the four phases, the second factor tends to balance the individuals between the four phases ($\#IndPhaseMax$ is the number of individuals representing the phase corresponding to the $argmax$ of the first factor and $\#Ind$ is the total number of different individuals in the population) and the third factor is a parsimony factor in order to avoid large structures. $NbMaxNodes$ has been experimentally tuned, and is currently fixed to 15.

4.2 Sharing Distance

The set of measurements $\{F_1, F_2, F_3, F_4\}$ provides a simplified representation in $\mathbb{R}^4$ of the discriminant capabilities of each individual. As the aim of a Parisian evolution is to evolve distinct subpopulations, each being adated to one of the four subtasks (i.e. characterize one of the four phases), it is natural to use an euclidean distance in this four dimensional phenotype space, as a basis of a simple fitness sharing scheme [7].

4.3 Aggregation of Partial Solutions and Global Fitness Measurement

At each generation, the population is shared in 4 classes corresponding to the phase each individual characterises the best (i.e. the argmax of $\max\{F_1, F_2, F_3, F_4\}$ for each individual). The 5% best of each class are used via a voting scheme to decide the phase of each tested sample[2]. The global fitness measures the proportion of correctly classified samples:

$$GlobalFit = \frac{\sum_{i=1}^{learningset} CorrectEstimations}{\#Samples}$$

The global fitness is then distributed as a multiplicative bonus on the individuals who participated in the vote:

$$LocalFit' = LocalFit \times (GlobalFit + 0.5)^{\alpha}$$

As $GlobalFit \in [0,1]$, multiplying by $(GlobalFit + 0.5) > 1$ corresponds to a bonus. The parameter α varies along generations, for the first generations (a third of the total number of generations) $\alpha = 0$ (no bonus), and then α linearly increases from 0.1 to 1, in order to help the population to focus on the four peaks of the search space.

Two sets of indicators are computed at each generation (see section 5, third line of figure 2):

[2] This scheme may also yield a confidence level of the estimation. This measurement is not yet exploited but can be used in future developments of the method.

- the sizes of each class, that show if each phase is equally characterised by the individuals of the population.
- the discrimination capability of each phase, computed on the 5% best individuals of each class as the minimum of:

$$\Delta = \max_{i \in [1,2,3,4]} \{F_i\} - \frac{\sum_{k \neq argmax\{F_i\}} \{F_k\}}{3}$$

5 Experimental Analysis

Available data have been collected from 16 experiments during 40 days each, yielding 575 valid measurements.[3] The derivatives of pH, la, *Km* and *Ba* have been averaged and interpolated (spline interpolation) for some missing days. Logarithms of these quantities are considered.

The parameters of both GP methods are detailed in table 2. The code has been developed in Matlab, using the GPLAB toolbox[10]. Comparative results of the four considered methods (multilinear regression, Bayesian network, GP and Parisian GP) are displayed in figure 1, and a typical GP run is analysed in figure 2.

Table 2. Parameters of the GP methods

	GP	**Parisian GP**
Population size	1000	1000
Number of generations	100	50
Function set	arithmetic and logical functions	arithmetic functions only
Sharing	no sharing	$\sigma_{share} = 1$ on the first third of generations, then linear decrease from 1 to 0.1 $\alpha_{share} = 1$ (constant)

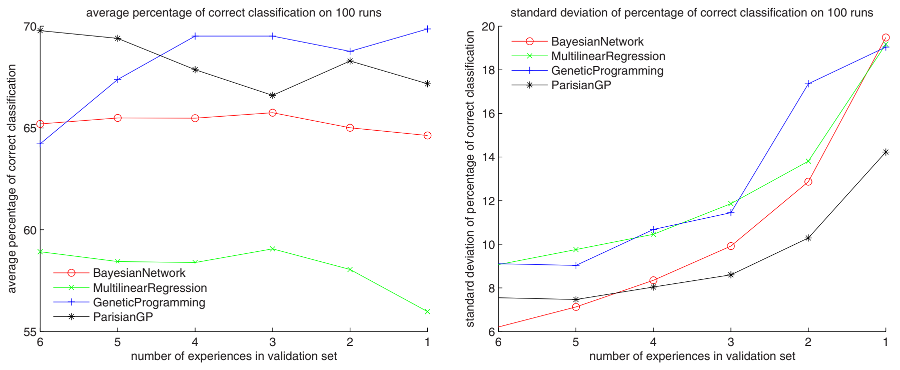


Fig. 1. Average (left) and standard-deviation (right) of recognition percentage on 100 runs for the 4 tested methods, the abscissa represent the size of the test-set

[3] The data samples are relatively balanced except for phase 3, which has a longer duration, thus a larger number of samples: We got 57 representatives of phase 1, 78 of phase 2, 247 of phase 3 and 93 of phase 4.

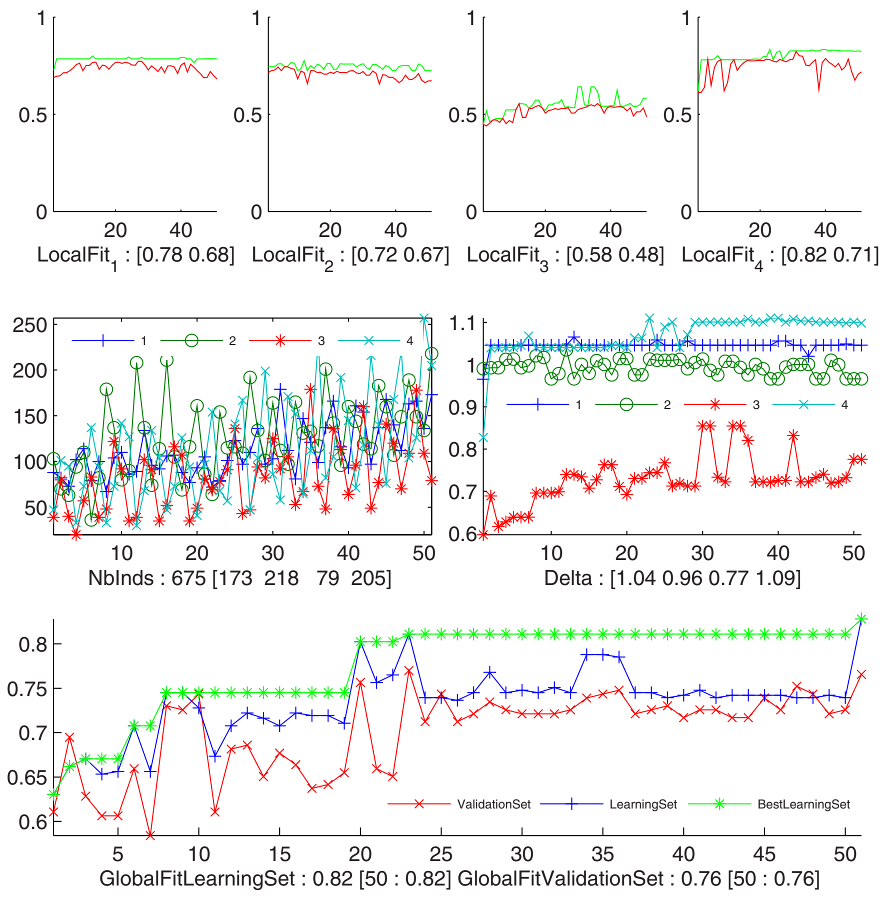


Fig. 2. A typical run of the Parisian GP:
- **First line:** the evolution with respect to generation number of the 5% best individuals for each phase: the upper curve of each of the four graphs is for the best individual, the lower curve is for the "worst of 5% best" individuals.
- **Second line left:** the distribution of individuals for each phase: the curves are very irregular but numbers of representatives of each phases are balanced.
- **Second line right:** discrimination indicator, which shows that the third phase is the most difficult to characterize.
- **Third line:** evolution of the recognition rates of learning and verification set. The best-so-far recognition rate on learning set is tagged with a star.

The multilinear regression algorithm used for comparison works as follows: the data are modeled as a linear combination of the 4 variables:

$$\widehat{Phase(t)} = \beta_1 + \beta_2 \frac{\partial pH}{\partial t} + \beta_3 \frac{\partial la}{\partial t} + \beta_4 \frac{\partial Km}{\partial t} + \beta_5 \frac{\partial Ba}{\partial t}$$

The 5 coefficients $\{\beta_1, \ldots, \beta_5\}$ are estimated using a simple least square scheme.

Experiments show that both GP outperform multilinear regression and Bayesian network approaches in terms of recognition rates. Additionally the analysis of a typical GP run (figure 2) shows that much simpler structures are evolved: The average size of evolved structures is around 30 nodes for the classical GP approach and between 10 an 15 for the Parisian GP.

It has also to be noted in figure 2 that co-evolution is balanced between the four phases, even if the third phase is the most difficult to characterize (this is in accordance with human experts' judgement, for which this phase is also the most ambiguous to characterize).

6 Conclusion and Future Work

This work is a first step toward the use of GP to model complex interactions within a cheese ripening industrial chain. Preliminary results presented in this paper show the effectiveness of GP schemes to capture subjective mechanisms related to human expertise. This point is extremely important for the automation of industrial process as well as for the transmission of expert knowledge.

Additionally, the developement of a cooperative-coevolution GP scheme (Parisian evolution) seems very attractive, as it allows to evolve simpler structure during less generations, and yield results that are easier to interpret. There are however some difficulties to overcome in future developments. First, the computation time is almost equivalent between both presented methods (100 generations of a classical GP against 50 generations of a Parisian one), as one "Parisian" generation necessitates more complex operations, all in all). One can expect a more favourable behaviour of the Parisian scheme on more complex issues than the phase prediction problem, as the benefit of splitting the global solutions into smaller components may be higher and may yield computational shortcuts (see for example [5]). The second difficulty comes from the fact that the Parisian sheme has to be adapted to the problem, it is not obvious for the moment that a convenient sub-problem splitting can be built for other, more complex, prediction problems.

References

1. Aldarf, M., Fourcade, F., Amrane, A., Prigent, Y.: Substrate and metabolite diffusion within model medium for soft cheese in relation to growth of Penicillium camembertii. J. Ind. Microbiol. Biotechnol. 33, 685–692 (2006)
2. Bongard, J., Lipson, H.: Active Coevolutionary Learning of Deterministic Finite Automata. Journal of Machine Learning Research 6, 1651–1678 (2005)
3. Barile, D., Coisson, J.D., Arlorio, M., Rinaldi, M.: Identification of production area of Ossolano Italian cheese with chemometric complex aproach. Food Control 17(3), 197–206 (2006)
4. Baudrit, C., Wuillemin, P.-H., Sicard, M., Perrot, N.: A Dynamic Bayesian Metwork to represent a ripening process of a solf mould cheese (submitted)
5. Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Polar IFS + Parisian Genetic Programming = Efficient IFS Inverse Problem Solving. Genetic Programming and Evolvable Machines Journal 1(4), 339–361 (2000)

6. Davis, L.: Adapting Operators Probabilities in Genetic Algorithms. In: 3rd ICGA conference, pp. 61–69. Morgan Kaufmann, San Francisco (1989)
7. Deb, K., Goldberg, D.E.: An investigation of niche and species formation in genetic function optimization. In: Proceedings of the third Conference on Genetic Algorithms, pp. 42–50 (1989)
8. Ellis, D.I., Broadhurst, D., Goodacre, R.: Rapid and quantitative detection of the microbial spoilage of beef by Fourier transform infrared spectroscopy and machine learning. Analytica Chimica Acta 514(2), 193–201 (2004)
9. Gripon, A.: Mould-ripened cheeses. In: Fox, P.F. (ed.) Cheese: Chemistry, Physics and Microbiology, vol. 2, pp. 111–136. Chapman & Hall, London (1993)
10. Silva, S.: GPLAB A Genetic Programming Toolbox for MATLAB, `http://gplab.sourceforge.net/`
11. Ioannou, I., Mauris, G., Trystram, G., Perrot, N.: Back-propagation of imprecision in a cheese ripening fuzzy model based on human sensory evaluations. Fuzzy Sets And Systems 157, 1179–1187 (2006)
12. Ioannou, I., Perrot, N., Curt, C., Mauris, G., Trystram, G.: Development of a control system using the fuzzy set theory applied to a browning process - a fuzzy symbolic approach for the measurement of product browning: development of a diagnosis model - part I. Journal Of Food Engineering 64, 497–506 (2004)
13. Ioannou, I., Perrot, N., Mauris, G., Trystram, G.: Development of a control system using the fuzzy set theory applied to a browning process - towards a control system of the browning process combining a diagnosis model and a decision model - part II. J.Food Eng. (64), 507–514 (2004)
14. Jimenez-Marquez, S.A., Thibault, J., Lacroix, C.: Prediction of moisture in cheese of commercial production using neural networks. Int. Dairy J. 15, 1156–1174 (2005)
15. Leclercq-Perlat, M.N., Picque, D., Riahi, H., Corrieu, G.: Microbiological and Biochemical Aspects of Camembert-type Cheeses Depend on Atmospheric Composition in the Ripening Chamber. J. Dairy Sci. (89), 3260–3273 (2006)
16. Leclercq-Perlat, M.N., Buono, F., Lambert, D., Latrille, E., Spinnler, H.E., Corrieu, G.: Controlled production of Camembert-type cheeses. J. Dairy Res. (71), 346–354 (2004)
17. Ni, H.X., Gunasekaran, S.: Food quality prediction with neural networks. Food Technology 52, 60–65 (1998)
18. Ochoa, G., Lutton, E., Burke, E.: Cooperative Royal Road Functions. In: Evolution Artificielle, Tours, France, October 29-31 (2007)
19. Pinaud, B., Baudrit, C., Sicard, M., Wuillemin, P.-H., Perrot, N.: Validation et enrichissement interactifs d'un apprentissage automatique des paramètres d'un réseau bayésien dynamique appliqué aux procédés alimentaires, Journées Francophone sur les Réseaux Bayésiens, Lyon, France (2008)
20. Riahi, M.H., Trelea, I.C., Leclercq-Perlat, M.N., Picque, D., Corrieu, G.: Model for changes in weight and dry matter during the ripening of a smear soft cheese under controlled temperature and relative humidity. International Dairy Journal 17, 946–953 (2007)
21. Tarantilis, C.D., Kiranoudis, C.T.: Operational research and food logistics. Journal of Food Engineering 70(3), 253–255 (2005)

Readable and Accurate Rulesets with ORGA

Md Nor Ridzuan Daud and David Corne

School of Mathematics and Computer Sciences, Heriot-Watt University
Edinburgh EH14 8AS, UK
mnrd1@hw.ac.uk, dwcorne@macs.hw.ac.uk

Abstract. A key task for data mining is to produce accurate and descriptive models. `Human readable' models are often necessary to enable understanding, potentially leading to further insight, and also inducing trust in the user. Rules, or decision trees (if not too numerous or large) are readable, unlike, for example SVM models. However, descriptiveness and accuracy normally conflict; a challenge is to find algorithms that have both high accuracy and high readability. We introduce ORGA (Optimized Ripper using Genetic Algorithm) which hybridizes evolutionary search with the RIPPER ruleset algorithm. RIPPER is effective at producing accurate and readable rulesets, and we show that ORGA provides significant further improvement. ORGA outperforms overall a suitable set of comparative algorithms including implementations of RIPPER, C4.5 and PART. On a majority of the datasets, ORGA's outperformance of the other algorithms is spectacular, and it is rarely dominated in terms of both accuracy and readability.

Keywords: data mining, human readability, hybrid machine learning.

1 Introduction

There continues to be considerable research effort worldwide aimed at finding efficient and effective ways to learn predictive models from data. An ever-present issue in this field is the tradeoff between the accuracy of a model and its complexity. Broadly speaking, methods in data mining either produce complex models that are highly predictive but difficult to understand, or simple models that tend to have less favourable accuracy. An ongoing area of interest is the attempt to get the best of both worlds. That is, to find algorithms that produce models that are *both* accurate and readily understandable. An understandable model is `human-readable'; that is, the knowledge inherent in the model is easily grasped by (for example) a doctor. The most obvious and common such model is an ' IF...THEN' rule, or, more normally, a small set of such rules. Such models are advantageous in practice for several reasons. First, this kind of knowledge representation is a convenient way for experts to convey their knowledge to others. It follows that this is also an effective way to convey knowledge *to* the expert. Meanwhile, such rules are modular, each defining a relatively small and, at least in principle, independent piece of knowledge. Additionally, we make comparisons with algorithms that produce decision trees, which also come into the category of readable and understandable models.

There is therefore continued interest in algorithms that generate simple rules, since these are naturally and immediately readable. Typically, the better-performing such algorithms tend to be those that produce *sets* of rules. For such a ruleset, a rough idea of its readability is given by the number of rules in the set. The issue in this specific research area is therefore to find algorithms that tend to produce *small* and *accurate* rulesets. In some preliminary research and literature review we identified a number of high-performing algorithms in this respect, with RIPPER [1] being the dominant one in typical, overall performance. We therefore took RIPPER as the launch point from which to pursue further research towards improving the combined accuracy and readability of rulesets, interested particularly in medical datasets. RIPPER itself has two main phases, the second phase performing a particular kind of constrained optimization of the ruleset produced by the first phase. We expected that replacing this optimization stage with an evolutionary algorithm would lead to enhanced performance. This paper therefore presents the resulting algorithm, which we call ORGA (Optimized Ripper with GA), and evaluates its performance on a range of medical-theme datasets, in comparison to a selection of other ruleset generation algorithms, as well as some decision tree generation algorithms.

The remainder is set out as follows. Section 2 summaries relevant related work. In section 3 we outline the RIPPER algorithm and describe ORGA, which hybridizes RIPPER with an evolutionary algorithm. The experimental setup is described in section 4, which includes a summary of all of the datasets and algorithms compared, and an analysis of the results. Section 5 provides a concluding summary.

2 Background and Related Work

Data mining in medicine is considered by some machine learning communities as one of the more complex and problematic domains yet to be overcome [2]. Two of the key tasks in medicine are *diagnosis* and *prognosis*. To learn, express and convey knowledge for these tasks, rule-based representations of knowledge are well-placed, and physicians are ready to trust and believe the consequent diagnoses.

In an attempt to sample the state of the art as regards methods that combine accuracy and readability, in preliminary work we tested a wide range of algorithms available in the Weka toolkit on a collection of medical-themed datasets. The results were analysed based on broad measures of readability and accuracy, and we found that RIPPER ([1]; implemented in Java version as JRip inside Weka) stood out in its performance in these combined respects, and we chose this to therefore examine and build on RIPPER towards further progress in combining accuracy and readability.

RIPPER is quite a sophisticated algorithm (described further in the next section), and derives from a lineage of rule induction algorithms which started with Reduced Error Pruning (REP) [3]. The essence of REP was to repeatedly *grow* and *prune* a ruleset. Growing a ruleset simply means adding antecedents to a rule or adding a new rule in such a way as to maximize some aspect of performance, and pruning a rule means removing antecedents or rules, again in such a way that maximizes some (perhaps other) aspect of performance. E.g. when growing a rule, we may continually add antecedents to increase that maintain the rule's coverage of a particular class, while maintaining false positives below a given threshold. A key element of the REP algorithm was that the dataset was partitioned into separate sets for evaluating performance in the growing and pruning phases respectively. This led to rather good generalization performance. Further developments explored improvements the ‘grow and prune’ (also regarded as ‘overfit and simplify’) approach, until IREP [4] improved upon this approach further by building a ruleset one rule at a time. That is, the grow and prune strategy was first used to generate a single rule, and then, after removing from the training sets data covered by the ruleset so far, further rules are constructed by iterating the process.

RIPPER then emerged from IREP by making a number of sophistications and modifications, affecting the details of the heuristics and stopping criteria used in the grow and prune phases, and involving the addition of a ruleset optimization component (in itself using iterated grow and prune operators combined with a minimum description length (MDL) heuristic) to follow the (modified) IREP-constructed ruleset. The result was a highly efficient algorithm with excellent generalization performance.

The success and performance characteristics of RIPPER have led to a number of variants being explored; among these, relatively prominent in the machine learning community is PART, which hybridises elements of C4.5 [5] with RIPPER, and which is one of the algorithms used in our comparison set. Meanwhile the SLIPPER algorithm is a ‘boosted’ version of RIPPER [6], but we do not consider that further, since boosting approaches naturally lead to larger and hence less readable models.

In this paper we explore a variant of RIPPER in which the optimisation stage is replaced by an evolutionary search. Hybridisations of evolutionary search with other strategies for rule or decision tree learning are fairly frequent in the literature, e.g. Turney [7] introduced ICET, a hybrid of a genetic algorithm and a decision tree induction algorithm for cost-sensitive classification. ICET uses a genetic algorithm to evolve a population of biases for a modified version of C4.5), while Bala et al [8] proposed a hybrid learning methodology that integrates genetic algorithms and decision tree learning. In similar vein, Carvalho and Freitas [9] integrate adecision tree approach and a genetic algorithm. Meanwhile, Hsu et al. [10] proposed a hybrid of the association rule algorithm *Apriori* and genetic algorithms called AGA to discover a classification tree. Apriori is adopted to obtain useful clues based on which the GA is able to proceed its searching tasks in a more efficient way.

Several such hybrid approaches have been explored, together with a long and increasing literature of ‘pure’ (or mainly) evolutionary algorithm based approaches (e.g. see [11]), but there are actually very few explorations of approaches where the evolutionary component is ‘light’, in the sense that a carefully designed and effective heuristic-based algorithm, such as RIPPER, is modified a little to contain an evolutionary algorithm By adopting this approach first, we expected to retain many of the strengths of RIPPER in generalisation ability, but by performing a less restrained optimisation perhaps enabling further improvements.

3 RIPPER and ORGA

The broad approach used in the original RIPPER algorithm is as follows. RIPPER builds a ruleset by repeatedly adding rules to an empty ruleset until all positive examples are covered. This naturally leads to a set of rules that covers all positive examples as well as many negative examples. Rules are then refined by greedily adding conditions to the antecedent of rules (starting with the empty antecendent) until no negative examples are covered. After a ruleset is constructed, an optimization stage then amends the ruleset so as to reduce its size and improve its fit to the training data. A combination of cross validation and minimum description length (MDL) heuristics are used to reduce the degree of data overfitting. RIPPER thus comprises two parts: ruleset-building, and ruleset-optimization. In the optimization stage, further grow and prune strategies are applied, with further heuristics and criteria to guide the search. Limited space prevents a complete description, but readers will find a full description in [1] and may inspect the Weka implementation of RIPPER (JRip).

In ORGA, we replace the optimization stage with an evolutionary search. Otherwise the algorithm is precisely the same as (the JRip implementation of) RIPPER. The task is to find an improved ruleset, given as a starting point the ruleset that emerged from the initial stage of RIPPER. The encoding is as follows. Each rule is a sequence of conditions {C1, C2, … Cn}, whose interpretation is "IF every condition is true, then the predicted class is X". X is always clear from context. In a standard way, each condition Ci is characterized by a triple {Ai, Ri, Vi}, which are respectively the attribute index, relational operator, and value. A chromosome is simply a variable length sequence of such triples. For example, the following chromosome:

3, <, 85, 2, >, 0, 8, =, "blue", 25,<,2

encodes the condition: "attribute 3's value is <85, attribute 2's value > 0, attribute 8's value is "blue" and attribute 25's value is < 2..". Naturally, each attribute value gene is an integer indexing the attributes, constrained to not index the attribute whose class is to be predicted. The relational operator gene is binary, indicating either "=" or "≠" for a categorical attribute, or "≤" or "≥" for a numerical attribute. Finally, the value gene contains a value within the range of values of the attribute in question; this is either a real value for a numerical attribute, or a category for a categorical attribute. Actually in this paper we only explore numerical datasets and hence encode only numerical attributes, mainly because almost all medical-theme datasets in the UCI repository are all-numeric. A ruleset is simply encoded as a set of such structures. The evolutionary algorithm component of ORGA then proceeds to evolve rulesets (i.e. a chromosome represents a single ruleset as in the Michigan approach), and fitness is simply its predictive accuracy.

Mutation operates by randomly selecting a condition from a randomly chosen rule, and making a random valid change to either its attribute, operator or value (for example, if the attribute is changed, then a random value will be selected from the existing values in the training set for that attribute). The overall algorithm for this optimization component is simply as follows. First, construct a single chromosome directly from the collection of rulesets emerging from the first stage of RIPPER. Then, construct a population of 100 rulesets by generating 100 mutated copies of the initial ruleset. Then we run a simple, mutation-only steady-state binary tournament selection evolutionary algorithm for 100 iterations. Hence, the additional runtime involved with ORGA is quite negligible; also, ORGA is unoptimised in terms of its parameters and strategies, representing one of our first attempts to see if RIPPER could be improved by using a basic evolutionary approach.

4 Datasets, Algorithms, Experimental Setup and Results

4.1 Benchmark Datasets Used in This Study

The datasets used in this investigation were obtained from the UCI Machine Learning Repository. Considering our interest in readability, which is a particular concern for data mining applications in medicine, we concentrate on medical datasets. At the time of writing, we consider only datasets where all attributes are numerical, and this is the case for 9 of the 10 datasets in the UCI repository that have a medical theme. A summary of these datasets is in Table 1.

Table 1. UCI Repository medical-theme datasets used in this study

No.	Dataset	Citation	Instances	Attributes	Classes
1	Breast-cancer (Wis-	[12]	569	30	2
2	echocardiogram	[13]	132	9	2
3	heart-disease	[14]	303	13	5
4	hepatitis	[15]	155	19	2
5	lymphography	[16]	148	18	4
6	pima-diabetes	[17]	768	8	2
7	primary-tumor	[16]	339	18	19
8	thyroid-gland	[18]	215	5	3
9	Horse-colic	[19]	300	23	2

4.2 Ruleset and Decision Tree Algorithms Used for Comparative Experiments

Exploiting the Waikato Environment for Knowledge Analysis (Weka [20]), we chose to compare with algorithms from the Weka toolkit that produce readable models (i.e. rulesets or decision trees), and can be applied directly to datasets with *both* numeric and nominal attributes and can handle several (i.e. more than two) predictive classes. Although we only use datasets in this study with all-numeric datasets, our ultimate interest is in techniques than can handle mixed-attribute datasets. This resulted in the set of algorithms summarized in Table 2, for which we employed the Weka implementations with default parameters.

OneR or “One Rule” is a simple algorithm proposed by Holt [21] which builds one rule for each attribute in the training data and then selects the rule with the smallest error rate. Ridor is the RIpple-DOwn Rule learner proposed by Gaines and Compton [22]. It generates a default rule first and then the exceptions for the default rule with the least (weighted) error rate. Then it generates the "best" exceptions for each exception and iterates until pure. Exceptions are a set of rules that predict classes other than the default. IREP [4] is used to generate the exceptions. PART is a separate-and-conquer rule learner proposed by Frank and Witten [23], which produces ‘decision lists’. PART builds a partial C4.5 decision tree in each iteration and makes the "best" leaf into a rule. The algorithm is a combination of C4.5 [5] and RIPPER [1]. JRip is the Weka implementation of RIPPER which, as we have seen, builds a ruleset by repeatedly adding rules to an empty ruleset until all positive examples are covered.

Table 2. Selected comparative algorithms

No.	Algorithm	Acronym	Classifier	Citation
1	OneR	OneR	Rule	[21]
2	Ridor	Ridor	Rule	[22]
3	PART	PART	Rule	[23]
4	JRip	JRip	Rule	[1, 20]
5	DecisionTable	DT	Rule	[24]
6	ConjunctiveRule	CR	Rule	[20]
7	J48 (C4.5)	J48	Tree	[5, 20]
8	DecisionStump	DS	Tree	[25]
9	RandomTree	RT	Tree	[20]
10	REPTree	RTree	Tree	[20]

DecisionTable builds a rule using a simple decision table majority classifier as proposed by Kohavi [24]. It summarizes the dataset with a 'decision table' which contains the same number of attributes as the original datase. It employs the wrapper method to find a good subset of attributes for inclusion in the table. ConjuctiveRule implements a single conjunctive rule learner and is described in [20]. A rule consists of antecedents "AND"ed together and the consequent (class value) for the classification/regression. In this case, the consequent is the distribution of the available classes (or mean for a numeric value) in the dataset. If the test instance is not covered by this rule, then it's predicted using the default class distributions/value of the data not covered by the rule in the training data. J48 is the weka implementation of C4.5 [5, 20]. DecisionStump builds simple binary decision 'stumps' (1 level decision tress) [25]. It is usually used in conjunction with a boosting algorithm such as LogitBoost [26]. RandomTree considers *K* randomly chosen attributes at each node as described in [20]. It performs no pruning. Finally, REPTree is a fast decision tree learner [20] which builds a tree using information gain/variance and prunes it using reduced-error pruning.

4.3 Experimental Setup

Experiments were done to compare the performance of ORGA with each of the algorithms described in section 4.2, on each of the datasets noted in section 4.1. Each individual experiment, characterized by the pair {Algorithm, Dataset}, was set up as follows. When the algorithm was ORGA, a simple unoptimized parameter set was employed, as detailed in section 3. In all other cases, the standard default settings of Weka version 3.5 were used for the algorithm in question. 10-fold cross-validation was used, where the dataset was randomly partitioned into ten approximately equally sized portions, and the result of a single trial was the mean accuracy on the test sets over the ten folds. The entire process was repeated for ten independent trials; so that we have a sample of 10 performance estimates for each {Algorithm, Dataset} pair. Finally, we note estimates of readability for each algorithm as follows. For algorithms that generate rulesets, we record the mean value of the number of rules in the final ruleset of each fold, and record the mean of this over the 10 trials. For algorithms that generate decision trees, we record the number of non-leaf nodes in the tree as an analogous measure. Table 3 provides the raw results, recording for each {Algorithm, Dataset} pair the mean (upper) and standard deviation (lower) of the ten trials.

Table 3. Raw results for various rule and tree algorithms and ORGA. Boldface indicates (equal) best mean value among the algorithms.

	Ruleset algorithms							Decision Tree algorithms			
Dataset	OneR	Ridor	PART	JRip	DT	CR	ORGA	J48	DS	RT	REPTree
Breast	88.2	93.7	**94.4**	93.8	93.3	90.28	92.0	94.1	88.6	91.0	93.4
	0.89	0.41	0.60	0.59	0.57	0.45	2.99	1.06	0.66	0.77	0.56
Echo	92.2	90.0	86.6	90.8	91.8	92.0	**92.4**	90.0	**92.4**	77.1	91.3
	0.32	1.85	1.31	0.72	0.37	0.40	1.22	1.27	0.0	2.04	0.97
Heart	51.4	54.4	51.1	54.5	56.6	53.9	**67.6**	52.6	51.8	48.5	56.2
	0.82	1.65	2.01	1.16	1.46	1.16	3.95	1.40	0.94	1.95	1.24
Hepatitis	82.5	78.6	79.9	79.0	79.4	78.8	**86.9**	79.2	77.9	78.8	78.8
	0.90	1.76	2.05	1.61	2.42	1.44	1.89	1.18	1.36	3.27	2.00
Lymph	75.7	76.8	76.3	**78.8**	75.7	65.1	77.7	78.5	65.9	73.1	75.3
	0.0	4.60	2.17	2.72	1.23	2.90	5.6	1.22	2.33	4.25	2.78
Pima	71.7	73.2	73.7	74.1	73.9	70.6	**76.6**	73.9	71.4	66.8	73.7
	0.68	0.82	1.19	1.43	0.67	1.05	1.2	1.36	0.83	1.37	1.4
p-tumor	27.0	37.6	40.4	38.6	38.1	27.8	**47.0**	41.3	27.14	33.1	39.9
	0.37	1.74	0.85	0.98	1.22	1.01	3.64	1.76	0.0	1.60	1.01
thy-g	92.4	93.5	**95.0**	93.3	92.5	78.7	94.0	93.8	76.8	91.2	92.4
	0.59	0.82	0.96	0.79	0.71	0.81	2.64	1.17	0.72	1.44	0.59
horse-c	69.4	70.4	72.1	71.5	72.5	65.0	**78.5**	73.0	73.0	65.5	72.3
	1.62	1.71	1.46	1.21	0.89	2.87	1.98	0.90	0.0	1.96	1.08

Table 4. Mean rough measures of readability for various rule and tree algorithms and an unoptimsed implementation of ORGA. In this case, boldface indicates algorithms whose mean accuracy was better than that of ORGA, enabling us to contrast the rough relative readability of ORGA in this circumstance.

	Ruleset algorithms							Decision Tree algorithms			
Dataset	OneR	Ridor	PART	JRip	DT	CR	ORGA	J48	DS	RT	REPTree
breast	4	**5**	**7**	5.4	**88**	1	5.8	**25**	3	111.2	**11.4**
	0	0.7	0	0.8	0	0	0.8	0	0	16.0	2.8
echo	3	2.9	6	2.3	3	1	2.7	3	**3**	72.8	3
	0	1.5	0	0.5	0	0	0.8	0	0	22.3	0
heart	4	41.7	49	4	8	1	13.2	67	3	326.2	22.4
	0	12.8	0	1.2	0	0	1.2	0	0	1.9	5.6
hepatitis	3	2.5	8	2.7	46	1	2.3	21	3	135.8	8.2
	0	0.5	0	0.8	0	0	1.5	0	0	14.6	2.2
lymph	3	8.7	11	**6.7**	30	1	5.5	**25**	3	143.2	13.4
	0	3.1	0	1.2	0	0	1.8	0	0	10.3	3.6
pima	8	6.3	13	4.1	32	1	5.7	39	3	496	31.8
	0	1.8	0	0.9	0	0	1.4	0	0	23	6.2
p-tumor	0	331.4	46	8.8	53	1	15	81	3	487.6	34.6
		111.5	0	0.8	0	0	2.5	0	0	28.1	8.2
thy-g	3	8.3	**4**	5	21	1	5.6	17	3	45.2	8.8
	0	1.7	0	0.7	0	0	0.5	0	0	8.2	1.8
horse-c	3	5.2	15	3.9	24	1	2.4	11	3	538.8	20
	0	1.4	0	0.9	0	0	0.8	0	0	25.5	5.3

Table 4 summarises the results in terms of the rough estimate of readability provided by mean number of rules in a ruleset, or non-leaf nodes in the tree, in the best-performing model delivered by each trial; again the mean and standard deviation are provided in each case. These are very rough measures of readability, and we aim to investigate refined measures in ongoing work. For now, they will serve to provide a usable guide as to the readability/accuracy tradeoffs among these algorithms. In particular, Table 4 highlights those cases in which other algorithms performed better than (or as well as) ORGA in terms of mean accuracy. This happened in the case of the breast-cancer, echo-cardiogram, and lymphography and thyroid-gland datasets. Considering cases of ruleset algorithms, we can see that ORGA produced equally smal rulesets than the slightly more accurate rivals in 4 of these 5 cases. Meanwhile, ORGA always produced a reasonably small number of rules. A remarkable result is the case of the hepatitis dataset, in which ORGA often produced a single rule, while outperforming all other algorithms in terms of mean result. Finally, in Table 5 we summarise findings in terms of statistical significance. For each dataset, we compare the ten results from ORGA with (individually) the ten results of each of the other algorithms using a randomization test [27], yielding *p* values for one-tailed hypotheses based on the difference in means; the table summarises the significance results for each dataset.

Table 5. Based on randomisation tests, summary of the statistical significance of apparent differences between ORGA and other algorithms for each dataset

Dataset	Statistical significance notes
Breast	PART ($p = 0.003$) and j48 ($p = 0.027$) outperform ORGA with confidence level >= 95%; JRip outperforms ORGA with >90% confidence ($p = 0.052$). REPTree and Ridor show no significant difference to ORGA, while ORGA is statistically superior to the remaining algorithms.
Echo	ORGA is not statistically different from OneR or DS, but is superior to DT with confidence >90% ($p = 0.087$), CR with confidence >90% ($p = 0.08$), and all others.
Lymph	No algorithm is statistically superior to ORGA with confidence >=95%
thy-g	PART is statistically superior to ORGA with $p = 0.048$; no significant difference between ORGA and J48 or ORGA and Ridor; ORGA superior to all other algorithms
Heart, hepatitis, pima,p-tumour, horse-c	ORGA superior to all others with p values < 0.001

5 Summary and Observations

ORGA is clearly the overall best, with outstandingly better performance than all other tested algorithms on five of the nine datasets, and only outperformed occasionally by PART, and once by JRip. ORGA clearly has desirable readability properties. In this context, the algorithms tend to fall into three categories. CR, DS and OneR provide readable models by design (e.g. CR produces precisely a single conjunctive rule), and tend to underperform in terms of accuracy. In a second category are DT, J48 (C4.5) and RT, which are more concerned with accuracy than readability; performance tends to be competitive, but at the expense of readability. Finally, Ridor, PART, JRip, REPTree and ORGA attempt to do well at both accuracy and readability. Ridor and REPTree are less successful in terms of accuracy and unstable in terms of readability;

meanwhile, PART, JRip and ORGA are good performers all round. This is not surprising considering that each of these is a fairly sophisticated hybrid approach.

Clues that might help explain ORGA's overall excellent performance might be as follows. Where ORGA excels seems to be on the many-class datasets; meanwhile its worst performance, relatively speaking, was on the dataset with the most attributes. Naively (since based on just this small sample of datasets) this suggests that ORGA may be specialised for many-class few-attribute datasets. This idea is in harmony with the observation that the optimisation stage in ORGA involves a fixed, limited number of evaluations of new rules, which means increasingly less chance to explore the space of possible rulesets as the number of attributes increases. Meanwhile, RIPPER involves a series of stages that build rules specialised to each class in the dataset, in this way making a specific effort to do well on many-class problems. This is reflected in the fact that, as well as ORGA, the other algorithms based (at least partly) on RIPPER tend to do well on the >2-class datasets.

Ongoing work will examine these tentative hypotheses, towards a better understanding of how the attainment of both excellent accuracy and readability can be reliably achieved. In parallel and guided by this effort, we expect that additional performance gains are likely to be found as we explore alternative places in ORGA's parameter and strategy spaces.

References

[1] Cohen, W.W.: Fast effective rule induction. In: Machine Learning: Proceedings of the Twelfth International Conference, Lake Tahoe, California (1995)

[2] Cios, K.J., Moore, G.W.: Medical data mining and knowledge discovery: Overview of key issues. In: Cios (ed.) Medical Data Mining and Knowledge Discovery, pp. 1–20. Physica-verlag, New York (2001)

[3] Pagallo, G., Haussler, D.: Boolean feature discovery in empirical learning. Machine Learning 5(1) (1990)

[4] Furnkranz, J., Widmer, G.: Incremental Reduced Error Pruning. In: Cohen, W., Hirsh, H. (eds.) Proceedings of the 11th International Conference on Machine Learning (ML 1994), pp. 70–77. Morgan Kaufmann, New Brunswick (1994)

[5] Quinlan, R.: C4.5: Programs for Machine Learning. Kaufmann Publishers, San Mateo (1993)

[6] Cohen, W.W., Singer, Y.: A Simple, Fast and Effective Rule Learner (1999)

[7] Turney, P.D.: Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. JAIR 2, 369–409 (1995)

[8] Bala, J., Huang, J., Vafaie, H., DeJong, K., Wechsler, H.: Hybrid learning using genetic algorithms and decision tress for pattern classification. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995, Montreal, Canada, pp. 719–724 (1995)

[9] Carvalho, D., Freitas, A.: A hybrid decision tree/genetic algorithm method for data mining. Information Sciences 163(1-3), 13–35 (2004)

[10] Hsu, P.L., Lai, R., Chiu, C.C.: The hybrid of association rule algorithms and genetic algorithms for tree induction: an example of predicting the student course performance. Expert Systems with Applications 25(1), 51–62 (2003)

[11] Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer, Heidelberg (2002)

[12] Street, W.N., Wolberg, W.H., Mangasarian, O.L.: Nuclear feature extraction for breast tumor diagnosis. In: IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, San Jose, CA, vol. 1905, pp. 861–870 (1993)

[13] Kan, G., Visser, C., Kooler, J., Dunning, A.: Short and long term predictive value of wall motion score in acute myocardial infarction. British Heart Journal 56, 422–427 (1986)

[14] Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., Froelicher, V.: International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology 64, 304–310 (1989)

[15] Diaconis, P., Efron, B.: Computer-Intensive Methods in Statistics. Scientific American 248 (1983)

[16] Michalski, R., Mozetic, I., Hong, J., Lavrac, N.: The Multi-Purpose Incremental Learning System AQ15 and its Testing Applications to Three Medical Domains. In: Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 1041–1045. Morgan Kaufmann, Philadelphia (1986)

[17] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., Johannes, R.S.: Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proc. of the Symp. on Computer Applications and Medical Care, pp. 261–265. IEEE Computer Society Press, Los Alamitos (1988)

[18] Coomans, D., Broeckaert, M., Jonckheer, M., Massart, D.L.: Comparison of Multivariate Discriminant Techniques for Clinical Data - Application to the Thyroid Functional State. Meth. Inform. Med. 22(1983), 93–101 (1983)

[19] Dougherty, J., Kohavi, R., Sahami, M.: Supervised and Unsupervised Discretization of Continuous Features. In: Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, pp. 194–202. Morgan Kaufmann, San Francisco (1995)

[20] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

[21] Holt, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine Learning 11, 63–91 (1993)

[22] Gaines, B.R., Compton, P.: Induction of Ripple-Down Rules Applied to Modeling Large Databases. J. Intell. Inf. Syst. 5(3), 211–228 (1995)

[23] Frank, E., Witten, I.H.: Generating Accurate Rule Sets Without Global Optimization. In: Fifteenth International Conference on Machine Learning, pp. 144–151 (1998)

[24] Kohavi, R.: The Power of Decision Tables. In: 8th ECML, pp. 174–189 (1995)

[25] Witten, I.H., Frank, E., Trigg, L., Hall, M., Holmes, G., Cunningham, S.J.: Weka: Practical machine learning tools and techniques with java implementations. In: Proc. ICONIP/ANZIIS/ANNES 1999 Int. Workshop: Emerging Knowledge Engineering and Connectionist-Based Info. Systems, pp. 192–196 (1999)

[26] Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. The Annals of Statistics 28, 337–374 (2000)

[27] Edgington, E.S.: Randomization tests, 3rd edn. Marcel-Dekker, New York (1995)

A Distributed Memetic Algorithm for the Routing and Wavelength Assignment Problem

Thomas Fischer, Kerstin Bauer, and Peter Merz

Department of Computer Science, University of Kaiserslautern, Germany
{fischer,k_bauer,pmerz}@cs.uni-kl.de

Abstract. The Routing and Wavelength Assignment Problem deals with the routing of telecommunication traffic in all-optical networks. Extending existing algorithms, we present a memetic algorithm (MA) for the static RWA by introducing a recombination operator and a scheme for distributing the computation. Compared to previously achieved results for this problem, our MA significantly improves the solution quality. We find provably optimal results for previously unsolved problem instances. The distributed variant using epidemic algorithms allows to find solutions of quality comparable to the MA in less real-time.

1 Introduction

The *Routing and Wavelength Assignment* problem (RWA), an NP-complete [1] graph-theoretical problem, deals with *Wavelength Division Multiplexed* (WDM) optical networks, where communication requests between nodes in a network have to be fulfilled by routing them on optical fiber links with given capacities.

A problem instance of the RWA is a physical network represented by a graph $G = (V, E, W)$ with nodes V, edges E, and wavelengths W. The optical fiber links in the physical network are represented by E (here undirected) and on each links each wavelength in W is available. A node in V can be starting point u^r or end point v^r of a connection request $r = (u^r, v^r, d^r) \in R$ with a demand of $d^r \in \mathbb{N}^+$. For each unit of demand a *lightpath* has to be established. A lightpath is a path between two nodes in the physical network utilizing one wavelength on each link. The *wavelength continuity constraint* requires the path to use the same wavelength on every link, the *wavelength conflict constraint* states that no wavelength on a link may be used by more than one lightpath at the same time.

In the RWA's *static* variant, a set of requests is given and one can either minimize the number of wavelengths to route all requests or maximize the number of routed requests for a given set of wavelengths. In the *dynamic* variant, requests turn up over time and one has to maximize the number of routed requests. Here, we focus on minimizing the number of used wavelengths in the static case.

Next, related work and a previous publication is discussed. We present our MA and the recombination operator in Sec. 2. The transformation into a distributed MA is described in Sec. 3. After motivating our experimental setup in Sec. 4, we present our results in Sec. 5.

1.1 Related Work

Early papers were due to Bala *et al.* [2] and Chlamtac *et al.* [3], where both wavelength constraints were defined and algorithms for the dynamic RWA were presented. A popular approach for the RWA is to split the solution finding process into solving the subproblems of routing and wavelength assignment independently. For the static RWA, Banerjee and Mukherjee [4] used this approach, where the routing part is solved by relaxing the wavelength continuity constraint, solving a fractional multicommodity flow problem, and using randomized rounding to find paths in the graph. For the wavelength assignment a graph coloring problem is solved, where the nodes represent the paths and the edges state whether the incident paths share a common physical link. Another approach for solving the wavelength assignment part was given by Manohar and Shevgaonkar [5], who defined it as an instance of the maximum edge disjoint paths problem, where a maximum-sized subset of all paths in a graph is wanted holding that no two paths share a common link. Iteratively, among the paths with no wavelength assigned, the maximum subset is determined the next unused wavelength assigned to that path set, which is then removed from subsequent iterations. A general overview and classification of algorithms solving both subproblems independently is provided in [6] by Zang *et al.* and in [7] by Choi *et al.*

Sinclair [8] presented a memetic algorithm for the static RWA. Its intricate cost function does not match any of the RWA's objectives as stated above. The algorithm features mutation, recombination, and two local search operations which are applied with varying probabilities. The mutation reroutes a given request on a path randomly chosen from the set of k-shortest paths between the endpoints using the first available wavelength. The recombination performs a crossover where the set of requests is split and both offsprings contain paths and wavelength assignments from one of the halves while the other half only provides the paths using new wavelength assignments. The first local search tries to reroute a request in a wavelength from a high index in a lower-indexed wavelength using a k-shortest path. The second local search operates similarly, but here a target wavelength is chosen first and all conflicting paths are rerouted before rerouting the path from the high-indexed wavelength. Given that the algorithm uses problem-specific operators, the large population size (500) and the vast number of generations (100 000) question its efficiency.

Skorin-Kapov [9] introduced a construction heuristic called BFD_RWA based on bin-packing algorithms. Requests are sorted non-increasingly by length of their shortest paths and routed in the wavelength with shortest available path.

In [10], we presented a new set of benchmark instances, on which an iterated local search (ILS) algorithm was applied. The ILS consists of a local search (LS) and a mutation operator. The LS's idea is to move requests from less used wavelength to highly used wavelengths with the intention to clear already sparse wavelengths. The mutation operator randomly selects two wavelengths λ_1 and λ_2 where w. l. o. g. the load of λ_2 is smaller than λ_1. A path routed in λ_2 is randomly chosen and moved to λ_1 by using the shortest path between both endpoints. Any conflicting path in λ_1 is forcefully removed and later reinserted

```
function InitializePopulation(Graph $G$, Requests $R$)
    $S \leftarrow \emptyset$
    for $i \leftarrow 1, \ldots, n$ do
        $s \leftarrow$ BFD_RWA$(G, R)$
        $S \leftarrow S \cup \{s\}$
    return $S$

function MemeticAlgorithmRWA(Graph $G$, Requests $R$)
    $S \leftarrow$ InitializePopulation$(G, R)$
    while !TerminationReached do
        $S' \leftarrow \emptyset$
        for all $s \in S$ do
            if Random() $\leq$P[recomb] then
                $\overline{s} \leftarrow$ ChooseIndividual$(S, s)$
                $s' \leftarrow$ Recombinator$(G, s, \overline{s}, R)$
            else
                $s' \leftarrow$ Mutate$(s, R, strength)$
            $s'' \leftarrow$ LocalSearch$(s', R)$
            if $s'' < s$ then
                $S' \leftarrow S' \cup \{s''\}$
            else
                $S' \leftarrow S' \cup \{s\}$
        $S \leftarrow S'$
    return $S$
```

Fig. 1. Memetic algorithm for the RWA

by the construction heuristic's approach. The ILS provides solutions near the lower bound (determined by relaxing the wavelength continuity constraint and solving the resulting multicommodity flow problem) and, for some instances even optimal solutions can be achieved.

It is equivalent having between any node pair either at most one request with demand ≥ 1 or multiple requests with demand $= 1$ each. We use the latter definition to simplify the notation of algorithms presented in this paper.

2 Algorithms

In this paper, we present a recombination operator (Fig. 1) which is used to create a population-based memetic algorithm (MA) for the static RWA. Based on the ILS presented in [10], the MA adds the feature to handle populations of solutions and to recombine two solutions to one offspring solution.

The population is initialized by using the BFD_RWA construction heuristic for each individual. Although this heuristic orders requests by the length of their shortest path, requests are inserted in random order within each subset of equal shortest path lengths, thus resulting in different solutions. Until reaching some termination criterion the MA iterates over the population. Our termination criterion is an instance-dependent time limit allowing to compare different setups

```
1: function RECOMBINATOR(Graph $G$, Solution $a$, Solution $b$, Requests $R$)
2:     $s \leftarrow a$, $R' \leftarrow \emptyset$, $P_b \leftarrow \bigcup_{r \in R} p_{G,b}(r)$
3:     for $\lambda \in \{\lambda_1, \ldots, \lambda_{|W|} : u(\lambda_i) \geq u(\lambda_{i+1})\}$ do
4:         for $r \in R : \lambda_s(r) = \lambda$ do
5:             if $p_{G,s}(r) \in P_b$ then
6:                 $P_b \leftarrow P_b \setminus \{p_{G,s}(r)\}$
7:             else
8:                 $s \leftarrow s \setminus r$, $R' \leftarrow R' \cup \{r\}$
9:     for all $r \in R'$ do                          ▷ for all currently unrouted paths
10:        $\lambda_s(r) \leftarrow \arg\min_{\lambda \in W} \Psi_{G,s}(r, \lambda) \neq \emptyset$
11:        $p_{G,s}(r) \leftarrow \Psi_{G,s}(r, \lambda_s)$        ▷ route request on the first possible wavelength
12:    return $s$
```

Fig. 2. Recombination operator for the RWA

for the same instance. In each iteration, for each individual the same steps are performed: First, with given probabilities (Sec. 4) either a recombination (s.b.) or a mutation step as introduced in [10] is performed. The mutation strength can be varied, the actually used strategy is $10\%{\downarrow}_{2\%}$ (starting with mutation strength 10 % and decreasing in each iteration by 2 % until reaching the fixed minimum of 1 %) as in [11]. Once a provisional offspring has been created, this individual is improved by the LS used in [10].

For each individual of the next generation, offspring and parent are compared and the better one is kept. Thus, the only interaction between individuals of the same generation is the recombination operator. Setting the recombination probability to 0.0, the memetic algorithm equals to as many independent ILS runs as there are individuals in the population.

The recombination operator allows to combine common features of two parent individuals to one offspring. The offspring only keeps paths existing in both parents using the wavelength assignments from the better one. The remaining requests are later inserted using a construction heuristic. To determine the set of paths common to both parents, a matching problem between paths from both parents has to be solved. Two paths can be matched iff they use the same set of links regardless of the wavelengths assigned to the paths.

Details of the recombination operator are given in Fig. 2. W. l. o. g. parent solution a is better than b as defined by the length-lex ordering from [10]. The offspring solution s is initialized as a copy of a. To determine a matching between paths from a and b, b's set of paths is stored in the multiset P_b. In the recombination's first phase the algorithm iterates on the set of wavelengths sorted non-increasingly by usage. For each request using the current wavelength, it is checked whether the request's path $p_{G,s}(r)$ is element of P_b. If the test holds, a match has been found and the path is removed from P_b to prevent future matchings. Otherwise, the request is removed from s (removed requests are collected in R'). The sorting of wavelengths here keeps paths in already highly used wavelengths more likely than paths in less often used wavelengths. We argued that introducing a gap in already highly used wavelengths by not matching paths in

```
1: function DistributedMemeticAlgorithmRWA(Graph $G$, Requests $R$)
2:     $s \leftarrow$ InitializeIndividual($G, R$)
3:     while $\neg$TerminationReached do
4:         if $\{q_1, \ldots, q_k\} = Q \neq \emptyset$ then
5:             $\overline{s} \leftarrow q_1, \quad Q \leftarrow \{q_2, \ldots, q_k\}$
6:             $s' \leftarrow$ Recombinator($G, s, \overline{s}, R$)
7:         else
8:             $s' \leftarrow$ Mutate($s, R, strength$)
9:         $s'' \leftarrow$ LocalSearch($s', R$)
10:        if $s'' < s$ then
11:            $s \leftarrow s''$
12:        if Random$\leq$P[recomb] then
13:            SendToRandomNeighbor($s$)
14:    return $s$
```

Fig. 3. Distributed memetic algorithm for the RWA

those wavelengths can be less likely exploited for future improvements compared to paths in less often used wavelengths. Finally, the removed requests R' have to be reinserted into s using the concept of the BFD_RWA construction heuristic, where the request is routed in the first wavelength where a feasible path connecting both endpoints is available.

3 Distribution

The MA has been enhanced to a distributed memetic algorithm (DMA), differing from the MA by using populations of n individuals in one algorithm instance, n algorithm instances with one individual each operate independently. Resembling the MA, the DMA's instances exchange individuals regularly over the network using an epidemic algorithm [12].

The DMA is shown in detail in Fig. 3 is similar to the MA (Fig. 1), as the main differences are (a) the DMA operates on a single individual per algorithm instance (b) the exchange and recombination between algorithm instances is managed differently. Each algorithm instance has a receiving queue Q where incoming individuals are stored temporarily (realized by using an asynchronous thread). At the beginning of each iteration it is checked if the queue contains at least one element. If the test holds, the queue's top element is removed and used in a recombination operation. Otherwise, the current solution is mutated. After the obligatory local search and selection, with a given probability the current solution is sent to one randomly selected neighboring algorithm instance. The probability for sending an individual equals to the recombination probability in the original MA. Due to the asynchronous nature of the DMA and the queuing effects, the actual recombination probability may differ.

To build a neighborhood relation between algorithm instances (nodes) in distributed setups, we use an epidemic algorithm [12,13], where each node maintains a list of neighboring algorithm instances. For the node neighbor lists' initialization, one node is selected and its contact information is provided to other nodes. During the execution of the algorithm, from time to time each node chooses a neighbor and both nodes exchanges their neighbor lists. Any received list will be merged with the receiving node's list. The epidemic algorithm's membership protocol settled the neighbor list for each node in less than 5 s in each case providing a stable neighbor list. The receiving queue Q for incoming solutions was limited to 16 elements (twice the largest population size) and any solution received at a node with full queue was discarded.

4 Experimental Setup

In [10], problem instances based on network data from the SNDlib collection [14] were presented. As all but four of these instances have been optimally solved (lower bound reached) and for one the difference between best known solution and the lower bound is only one wavelength, we use the remaining instances `janos-us-ca`, `nobel-us`, and `zib54` for the experiments in this paper. For the SND problem, Atamtürk and Rajan [15] recently presented new benchmark instances, which we converted to RWA instances, too. As the authors use fractional demands in their data, we multiplied them by 100. Using the concept of undirected edges, we added the demands with different directions between the same nodes. For each instance size, the authors provide three networks: (a) average node degree 4 (b) average node degree 8 (c) edge density 75 %. We performed experiments with problem instances of size 15 and 20. The properties of all instances used in our experiments are summarized in Tab. 1.

We considered population sizes of 2, 4 and 8 and recombination rates of 0.2, 0.4, 0.6, 0.8, and 1.0 (recombination only). For comparison, the original ILS was reimplemented by using population size 1 and a recombination rate of 0.0

Table 1. Properties of instances used in our experiments 'Pairs' describes the number of node pairs communicating with each other, 'Requests' summarizes the demand (sum of paths to be established). 'LB' designates the known lower bound from [10] and 'UB' the best result found in any setup in this paper.

Instance	Nodes	Edges	Pairs	Requests	LB	UB	Time [s]
`janos-us-ca`[†]	39	122	1482	10173	1288	1288	1200
`nobel-us`	14	21	91	5420	670	670	300
`zib54`	54	81	1501	12230	705	705	600
`15.50.75`	15	90	72	9500	–	155	1500
`15.50.deg4`	15	48	72	9500	–	366	450
`15.50.deg8`	15	68	72	9500	–	258	450
`20.50.75`	20	150	137	18210	–	188	3000
`20.50.deg4`	20	82	137	18210	–	439	1000
`20.50.deg8`	20	106	137	18210	–	294	2000

[†] Instance has been modified, see text for details.

(no recombination). All experiments were repeated 30 times with different seeds. We used up to eight identical cluster PCs (Pentium 4 with 3 GHz running Linux) connected in a switched 100 MBit network. All software was written in Java.

Furthermore, to guarantee comparability between DMA and MA, both algorithms are given the same total time per instance (see Tab. 1) by dividing the time limit per CPU by the population size in the distributed case.

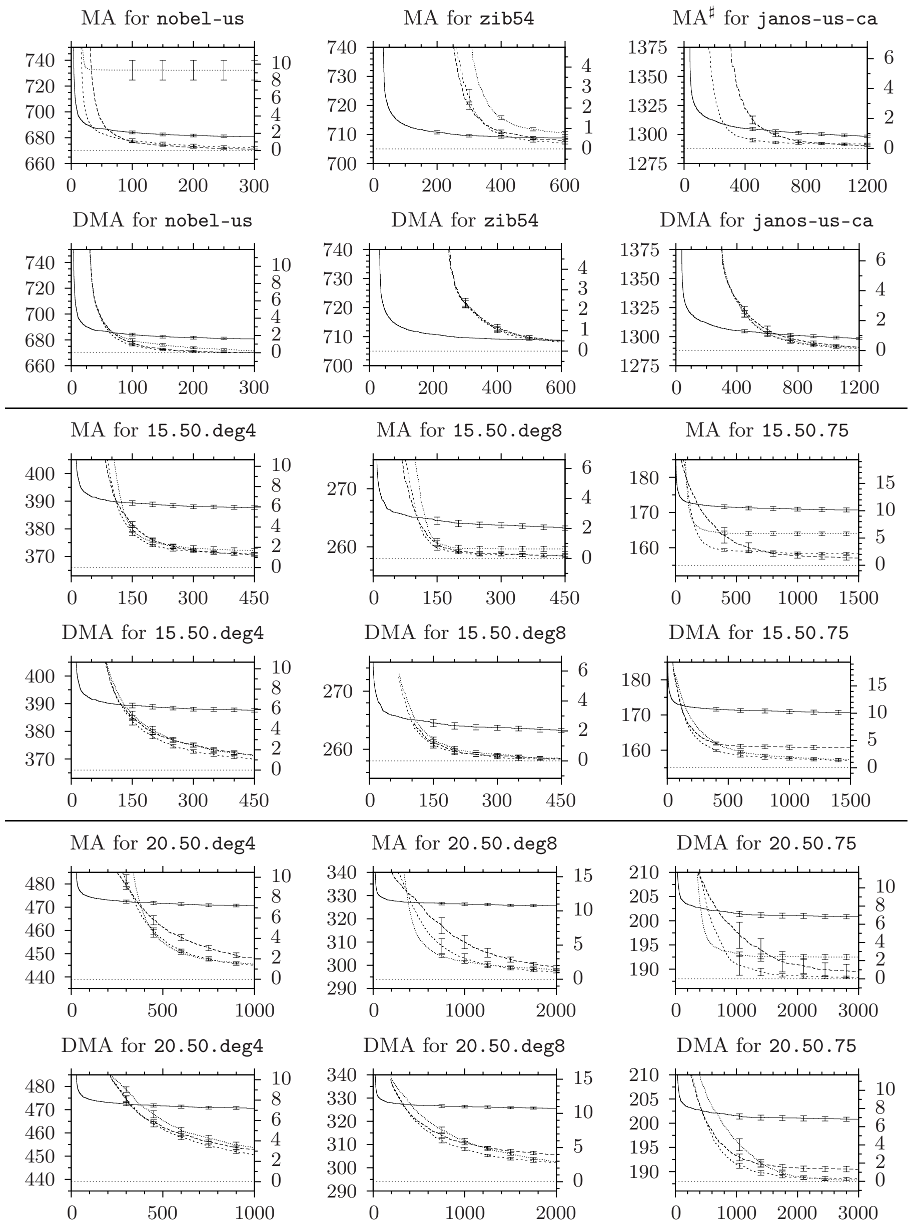
5 Experimental Results

Experiments using the MA and DMA were conducted as described above. In Fig. 4 the performance of both the distributed and non-distributed MA variants operating on 8 individuals for every benchmark instance is visualized. Except for instance `zib54` and a pathological case for `nobel-us` (discussed below), both variants of our MA find significantly better results compared to the original ILS. For the three instances already discussed in [10], our memetic algorithm was able to find optimal solutions for `janos-us-ca` in 4 MA setups and 93 DMA setups, for `nobel-us` in 28 and 402 setups, respectively, and for `zib54` in 36 and 82 setups, respectively, out of 450 runs each.

Regarding population size, small sizes are less capable of maintaining sufficient diversity within the population than larger populations. Interestingly, this effect is stronger for non-distributed than for distributed setups. The DMA's population stays more diverse, as the DMA's queue Q (see Fig. 3) provides a memory of older and thus more different solutions. E. g. for `20.50.deg4`, the percentage of paths with the same edges regardless of used wavelengths (*relative similarity* between two solutions) of both parents for a recombination in the MA is on average over time and all repetitions $> 98.5\,\%$ for all recombination rates, whereas for the corresponding DMA setup the similarity is $< 95.0\,\%$. For recombination rate 1.0 in the non-distributed case, the relative similarity decreases from 99.9 % to 95.5 % and 91.2 % for population sizes of 2, 4, and 8, respectively.

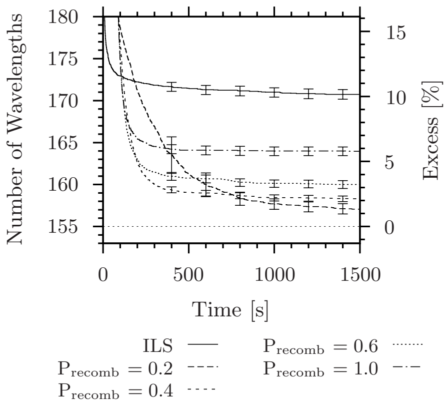
Regarding recombination rates, three different patterns can be observed: (a) all rates perform similarly (b) recombination rate 1.0 performs significantly worse (c) recombination rate 0.2 performs significantly worse. Case (b) occurs most often for the MA, whereas case (c) is more common for the DMA.

We discuss the setup with `15.50.75` (Fig. 5) with population size 8 in detail. The two plots show the run-time behavior of the non-distributed and distributed setup, respectively, for selected recombination rates in comparison to the original ILS and the best known solution. Furthermore, instance `15.50.75` represents recombination patterns (b) and (c), respectively. For each setup, the average number of generations, the effective recombination rate, and the average similarity of recombination partners are summarized below the plots. The number of generations decreases with increasing recombination rate as a recombination operation requires more computation time than a mutation operation. The effective recombination rate determined by counting the number of recombinations is exactly the expected value for the non-distributed case. For the DMA, the effective rates are lower as recombinations are only performed if an individual's queue Q

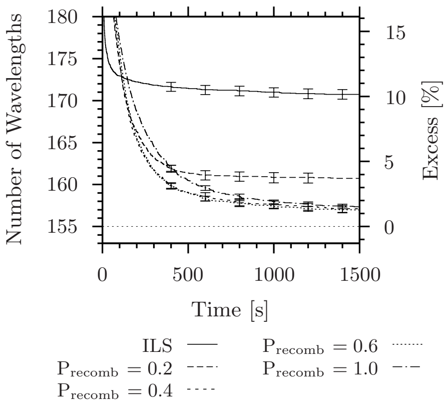


Line type – represents the ILS, types - -, ···, and ···· represent recombination rates 0.2, 0.4, and 1.0, respectively. The horizontal line at each plot's bottom is the best found solution's quality. 99 % confidence intervals are given.

Fig. 4. Overview on the performance for each instance both for the non-distributed and distributed variant operating on 8 individuals. Axis labels are the same as in Fig. 5. ♯ Setups with 1.0 recombination rate are out of scale.



(a) non-distributed MA



(b) Distributed MA

	Non-distributed				Distributed			
	0.2	0.4	0.6	1.0	0.2	0.4	0.6	1.0
# Gen.	2214.3	2317.6	2189.4	1959.8	1812.6	1619.6	1414.7	1151.2
$P^{\mathrm{eff}}_{\mathrm{recomb}}$	0.200	0.400	0.600	1.000	0.196	0.391	0.582	0.886
sim [%]	94.4 %	96.9 %	97.2 %	97.6 %	91.1 %	93.8 %	94.4 %	94.6 %

Fig. 5. Detailed analysis of experimental results for setups with instance `15.50.75` with 8 individuals. The horizontal line at each plot's bottom is the best found solution's quality. 99 % confidence intervals are given.

is non-empty. Due to varying time requirements for mutation and recombination and short-time effects, queues may be filled beyond the capacity limit for a few generations and thus incoming solutions get discarded. For high recombination rates, this effect is stronger explaining the lower effective recombination rates. Due to this effect pattern (b) does not occur for distributed setups.

In our recombination, parents pass on common path and wavelength assignments to their offspring thus requiring to insert new path and assignments to get a feasible solution (partial restart). Low similarity yields large partial restarts which may degrade the offspring's solution quality, for too high similarity the restart is too small to escape local optima. This model is well supported by our data as depicted in Fig. 5. In the non-distributed case, the final solution quality decreases with increasing similarity of the parents. In the distributed case, all recombination rates except rate 0.2 have the same parent similarity of about 94 % and the same solution quality. Only the recombination rate 0.2 has a significantly lower similarity and performs considerably worse.

6 Conclusions

In this paper we presented a memetic algorithm including a recombination operator for the static RWA which significantly improved former results. Furthermore, the MA can be efficiently distributed in a real network allowing to find results similar to the single CPU variant given the same total time summed over all participating CPUs. Future work will focus on combining our MA and DMA with the multilevel approach [11] for large instances.

References

1. Chlamtac, I., Ganz, A., Karmi, G.: Lightnet: Lightpath Based Solutions for Wide Bandwidth WANs. In: INFOCOM 1990, vol. 3, pp. 1014–1021 (1990)
2. Bala, K., Stern, T.E., Bala, K.: Algorithms for Routing in a Linear Lightwave Network. In: IEEE INFOCOM, vol. 1, pp. 1–9 (1991)
3. Chlamtac, I., Ganz, A., Karmi, G.: Lightpath Communications: An Approach to High Bandwidth Optical WANs. IEEE Trans. Comm. 40(7), 1171–1182 (1992)
4. Banerjee, D., Mukherjee, B.: A Practical Approach for Routing and Wavelength Assignment in Large Wavelength-Routed Optical Networks. IEEE J. Sel. Areas Comm. 14(5), 903–908 (1996)
5. Manohar, P., Manjunath, D., Shevgaonkar, R.K.: Routing and Wavelength Assignment in Optical Networks From Edge Disjoint Path Algorithms. IEEE Communications Letters 6(5), 211–213 (2002)
6. Zang, H., Jue, J.P., Mukherjee, B.: A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Optical WDM Networks. Optical Networks Magazine 1, 47–60 (2000)
7. Choi, J.S., Golmie, N., Lapeyrere, F., Mouveaux, F., Su, D.: A Functional Classification of Routing and Wavelength Assignment Schemes in DWDM networks: Static Case. In: OPNET 2000, pp. 1109–1115 (2000)
8. Sinclair, M.C.: Minimum cost routing and wavelength allocation using a genetic-algorithm/heuristic hybrid approach. In: Proc. 6th IEE Conf. Telecom. (1998)
9. Skorin-Kapov, N.: Routing and Wavelength Assignment in Optical Networks using Bin Packing Based Algorithms. EJOR 177(2), 1167–1179 (2007)
10. Bauer, K., Fischer, T., Krumke, S.O., Gerhardt, K., Westphal, S., Merz, P.: Improved Construction Heuristics and Iterated Local Search for the Routing and Wavelength Assignment Problem. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 158–169. Springer, Heidelberg (2008)
11. Fischer, T., Bauer, K., Merz, P.: A Multilevel Approach for the Routing and Wavelength Assignment Problem. In: Köppen, M., Raidl, G. (eds.) HEUNET 2008. IEEE Comp. Soc. Press, Los Alamitos (2008)
12. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic Algorithms for Replicated Database Maintenance. In: Schneider, F.B. (ed.) ACM PODC, pp. 1–12. ACM Press, New York (1987)
13. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based peer sampling. ACM Transactions on Computer Systems 25(3) (2007)
14. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0–Survivable Network Design Library. In: Proc. INOC 2007 (2007)
15. Atamtürk, A., Rajan, D.: Partition inequalities for capacitated survivable network design based on directed p-cycles. Discrete Optimization 5(2), 415–433 (2008)

Theoretical Analysis of Initial Particle Swarm Behavior

Sabine Helwig and Rolf Wanka

Department of Computer Science, University of Erlangen-Nuremberg, Germany
{sabine.helwig,rwanka}@informatik.uni-erlangen.de

Abstract. In this paper, particle trajectories of PSO algorithms in the first iteration are studied. We will prove that many particles leave the search space at the beginning of the optimization process when solving problems with boundary constraints in high-dimensional search spaces. Three different velocity initialization strategies will be investigated, but even initializing velocities to zero cannot prevent this particle swarm explosion. The theoretical analysis gives valuable insight into PSO in high-dimensional bounded spaces, and highlights the importance of bound handling for PSO: As many particles leave the search space in the beginning, bound handling strongly influences particle swarm behavior. Experimental investigations confirm the theoretical results.

1 Introduction

Particle Swarm Optimization (PSO) [1] is a population-based algorithm for global optimization. All population members, from now on called *particles*, explore the n-dimensional search space S of an optimization problem with objective function $f : S \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$. Without loss of generality (W.l.o.g.), we will assume minimization problems. Each particle has a *position* $\boldsymbol{x}_{i,t}$, a *velocity* $\boldsymbol{v}_{i,t}$, and a *fitness value* $f(\boldsymbol{x}_{i,t})$, where t is the iteration counter. A position $\boldsymbol{z}_1 \in S$ is called *better* than $\boldsymbol{z}_2 \in S$ iff $f(\boldsymbol{z}_1) < f(\boldsymbol{z}_2)$. The best search space position particle i has visited until iteration t is its *private guide* $\boldsymbol{p}_{i,t}$. To each particle, a subset of all particles is assigned as its neighborhood. The best private guide of all neighbors of particle i is called its *local guide* $\boldsymbol{l}_{i,t}$. In each iteration, position and velocity of each particle i are updated according to the following equations:

$$\begin{aligned}
\boldsymbol{v}_{i,t} &= \omega \cdot \boldsymbol{v}_{i,t-1} + c_1 \cdot \boldsymbol{r}_1 \odot (\boldsymbol{p}_{i,t-1} - \boldsymbol{x}_{i,t-1}) + c_2 \cdot \boldsymbol{r}_2 \odot (\boldsymbol{l}_{i,t-1} - \boldsymbol{x}_{i,t-1}) \\
\boldsymbol{x}_{i,t} &= \boldsymbol{x}_{i,t-1} + \boldsymbol{v}_{i,t}
\end{aligned}$$

where ω, c_1, and c_2 are prespecified parameters, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are vectors of random real numbers uniformly chosen between 0 and 1, and independently drawn every time they occur. $\odot$ denotes element-by-element vector multiplication.

In this paper, optimization problems with *boundary constraints* are studied, i.e., $S = [lb_1, ub_1] \times [lb_2, ub_2] \times \ldots \times [lb_n, ub_n]$ is bounded. W.l.o.g., we will assume $S = [-r, r]^n$. As many real-world problems and most benchmark suites have boundary constraints, a lot of strategies to handle them can be found in the literature (e.g., [2,3,4,5,6]). We will consider the following three strategies:

– *Infinity* allows particles to enter invalid space, does not alter positions or velocities, and skips the evaluation step for infeasible particles [6].
– *Absorb* sets invalid particles on the nearest boundary and all affected velocity components are set to zero [7].
– *Random* sets all invalid components of a particle's position vector to a random value, and the velocity is adjusted: $\boldsymbol{v}_{i,t+1} = \boldsymbol{x}_{i,t+1} - \boldsymbol{x}_{i,t}$ [4].

Most theoretical studies of PSO concentrate on the important task of selecting appropriate values for c_1, c_2, and ω [8,9,10]. Some widely-used standard settings were derived from these analyses, e.g., $c_1 = c_2 = 1.496172, \omega = 0.72984$ [6], or $c_1 = c_2 = 1.193, \omega = 0.721$ [11]. Often, PSO analyses assume 1-dimensional problems as each component of position and velocity vector is updated separately. However, for a deeper understanding of particle swarms in high-dimensional, bounded search spaces, the peculiarities of high-dimensional spaces have to be taken into account. Previous studies have already shown that the "curse of dimensionality", which means that high-dimensional spaces are not intuitive, is an important topic in particle swarm optimization [12]. It was proven that particles are initialized very close to at least one boundary. Moreover, it was shown that, *with overwhelming probability* (*w.o.p.*, for a definition, see Section 2), the best particle leaves the search space in the first iteration when velocities are initialized uniformly at random in $[-r, r]^n$.

In the following analyses, PSO in high-dimensional search spaces is studied in more depth. We will show that uniform velocity initialization causes not only the best, but all particles to leave the search space, w.o.p. The fact that many particles leave the search space was noted earlier [13,7], but never proven theoretically.

In order to avoid that too many particles leave the search space at the beginning, other velocity initialization strategies were proposed:

– *Zero* [13]: Particle velocities are initialized to zero.
– *Half-diff* [11]: Let $\boldsymbol{x}_{i,0}$ be the initial position of particle i, and $\boldsymbol{y}_i$ drawn uniformly at random in S. Then, $\boldsymbol{v}_{i,0}$ is set to $\frac{1}{2}(\boldsymbol{y}_i - \boldsymbol{x}_{i,0})$.

In Section 2, we will prove that using zero or half-diff initialization also causes many particles to leave the search space, w.o.p. We will derive some consequences for PSO application afterwards. In Section 3, different velocity initialization strategies and bound handling mechanisms will be studied experimentally on known benchmark problems.

2 Theoretical Results

Particle trajectories in the first iteration will be analyzed for three different velocity initialization strategies. Two main results will be derived:

– When using uniform velocity initialization, all particles leave the search space in the first iteration, w.o.p.

– When using zero or half-diff initialization, all particles which have a better neighbor than themselves leave the search space in the first iteration, w.o.p. For realistic optimization problems and most commonly-used neighborhood topologies, this is the majority of the particles.

The following assumptions will be used: $1 < c_2 \leq 2$, $0 < \omega \leq 1$, c_2 and ω do not depend on n, and particles are initialized uniformly at random in the n-dimensional search space $[-r, r]^n \subset \mathbb{R}^n$. The particles are connected via an arbitrary neighborhood topology, including the fully connected swarm. For each particle i with $\boldsymbol{x}_{i,0} \neq \boldsymbol{l}_{i,0}$, $\boldsymbol{l}_{i,0}$ is distributed uniformly at random in S. Actually, the local guides' positions depend on the optimization problem. However, the above assumption is not too restrictive for higher-dimensional problems, which is confirmed by Examples 1, 3, and 4 (*PSO Exp.*).

Definition: An event $A(n)$ happens *with overwhelming probability (w.o.p.)* with respect to n if there exists a constant $\gamma > 0$ such that $P(A(n)) = 1 - e^{-\Omega(n^\gamma)}$, where Ω belongs to the *big-O notation* for expressing asymptotic behavior. Hence, an overwhelming probability with respect to n rapidly approaches 1 when n increases.

2.1 Uniform Velocity Initialization

Theorem 1. *If velocities are initialized with uniform distribution in the search space, all particles which are initialized such that they have at least one neighbor with better fitness value than themselves (i.e., $\boldsymbol{x}_{i,0} \neq \boldsymbol{l}_{i,0}$) leave the search space, w.o.p.*

Proof. Let particle i be an arbitrary particle satisfying the above assumptions. As $\boldsymbol{p}_{i,0} = \boldsymbol{x}_{i,0}$, its position and velocity in the first iteration evaluate to

$$\begin{aligned} \boldsymbol{v}_{i,1} &= \omega \cdot \boldsymbol{v}_{i,0} + c_2 \cdot \boldsymbol{r}_2 \odot (\boldsymbol{l}_{i,0} - \boldsymbol{x}_{i,0}) \\ \boldsymbol{x}_{i,1} &= \boldsymbol{x}_{i,0} + \boldsymbol{v}_{i,1} = \omega \cdot \boldsymbol{v}_{i,0} + (\boldsymbol{1} - c_2 \cdot \boldsymbol{r}_2) \odot \boldsymbol{x}_{i,0} + c_2 \cdot \boldsymbol{r}_2 \odot \boldsymbol{l}_{i,0} \ . \end{aligned} \tag{1}$$

Hence, for fixed $\boldsymbol{r}_2$, the d-th component of $\boldsymbol{x}_{i,1}$, $x_{i,1,d}$, is the sum of three non-identical uniformly distributed, independent random variables. Its density function $f_{x_{i,1,d}}(z)$ can be computed using the formula presented by Bradley and Gupta [14, Theorem 1]. The probability that a particle crosses the boundary in dimension d evaluates to:

$$q_1(r_{2,d}, c_2, \omega) = \int_{-\infty}^{-r} f_{x_{i,1,d}}(z)\mathrm{d}z + \int_{r}^{\infty} f_{x_{i,1,d}}(z)\mathrm{d}z =$$

$$= \begin{cases} \frac{-3\omega^2+6c_2r_{2,d}\omega-4c_2^2r_{2,d}^2}{-12\omega(1-c_2r_{2,d})} (=: p_1) & \text{if } 0 \leq r_{2,d} < \frac{\omega}{2c_2} \\ \frac{\omega^2}{24(1-c_2r_{2,d})c_2r_{2,d}} (=: p_2) & \text{if } \frac{\omega}{2c_2} \leq r_{2,d} < \frac{2-\omega}{2c_2} \\ \frac{4c_2^2r_{2,d}^2+6\omega c_2r_{2,d}-8c_2r_{2,d}+3\omega^2+4-6\omega}{12\omega c_2r_{2,d}} (=: p_3) & \text{if } \frac{2-\omega}{2c_2} \leq r_{2,d} < \frac{2+\omega}{2c_2} \\ \frac{24+\omega^2+24c_2^2r_{2,d}^2-48c_2r_{2,d}}{-24c_2r_{2,d}(1-c_2r_{2,d})} (=: p_4) & \text{if } \frac{2+\omega}{2c_2} \leq r_{2,d} \leq 1 \end{cases} \tag{2}$$

As $r_{2,d}$ is uniformly drawn from $[0,1]$, the probability $p_A(c_2,\omega)$ that a particle violates the boundary in a specific dimension evaluates to:

$$p_A(c_2,\omega) = \int_0^1 q_1(r_{2,d},c_2,\omega)\mathrm{d}r_{2,d} =$$

$$= \begin{cases} (24\omega c_2)^{-1}\cdot(-36\omega+6\omega^2\ln(2)-12\omega^2\ln(2-\omega)+5\omega^2-36\omega\ln(2)+24\omega\ln(2-\omega)+8\ln(2) \\ \quad -16\ln(2-\omega)-3\omega^3\ln(\omega)+2\omega^3\ln(2-\omega)+8\ln(2+\omega)+12\omega\ln(2+\omega)+6\omega^2\ln(2+\omega) \\ \quad -24\omega\ln(c_2)-\omega^3\ln(c_2)+\omega^3\ln(2+\omega)+\omega^3\ln(c_2-1)+24\omega c_2) & \text{if } 2+\omega-2c_2<0 \\ (12\omega c_2)^{-1}\cdot(-10\omega+6\omega^2\ln(2)-6\omega^2\ln(2-\omega)+5\omega^2-12\omega\ln(2) \\ \quad +12\omega\ln(2-\omega)+8\ln(2)-8\ln(2-\omega)-\omega^3\ln(\omega)+\omega^3\ln(2-\omega) \\ \quad +4\ln(c_2)+6-6\omega\ln(c_2)+3\omega^2\ln(c_2)+6\omega c_2+2c_2^2-8c_2) & \text{if } 2+\omega-2c_2\geq 0 \end{cases} \tag{3}$$

Eq. (3) can be used for calculating $p_A(c_2,\omega)$ for specific values of c_2 and ω (see Example 1). In order to prove that particles leave the search space w.o.p., we must show that $p_A(c_2,\omega) > 0$. Therefore, we use the following fact:

$$p_A(c_2,\omega) = \underbrace{\int_0^{\frac{\omega}{2c_2}} p_1\mathrm{d}r_{2,d}}_{l_1(c_2,\omega)\geq 0} + \underbrace{\int_{\frac{\omega}{2c_2}}^{\frac{2-\omega}{2c_2}} p_2\mathrm{d}r_{2,d}}_{l_2(c_2,\omega)\geq 0} + \underbrace{\int_{\frac{2-\omega}{2c_2}}^{\min\{\frac{2+\omega}{2c_2},1\}} p_3\mathrm{d}r_{2,d}}_{l_3(c_2,\omega)\geq 0} + \underbrace{\int_{\min\{\frac{2+\omega}{2c_2},1\}}^{1} p_4\mathrm{d}r_{2,d}}_{l_4(c_2,\omega)\geq 0}$$

We compute $l_2(c_2,\omega) = \frac{\omega^2\cdot(\ln(2-\omega)-\ln(\omega))}{\ln(c_2)}$ and $l_1(c_2,1) = \frac{1+2\ln(2)}{24c_2} > 0$.

If $\omega < 1$, then $l_2(c_2,\omega) > 0$, otherwise $l_1(c_2,\omega) > 0$. Thus, $p_A(c_2,\omega) > 0$, and the probability that a particle leaves the n-dimensional search space is

$$p'_A(c_2,\omega,n) = 1-(1-p_A(c_2,\omega))^n = 1-e^{-\Theta(n)}\ . \tag{4}$$

□

Example 1. Two experiments were conducted for this and subsequent examples:

Conf. Exp.: In order to confirm that the mathematical analysis is correct under the given assumptions, the following experiment was conducted: $\boldsymbol{x}_{i,0}$, $\boldsymbol{v}_{i,0}$, $\boldsymbol{l}_{i,0}$, and $\boldsymbol{r}_2$ were randomly drawn according to the assumptions, and $\boldsymbol{x}_{i,1}$ was calculated according to Eq. (1). The probability for $\boldsymbol{x}_{i,1} \notin S$ was evaluated by performing 10^7 runs per considered problem dimensionality.

PSO Exp.: In order to determine the relevance of the theoretical results for PSO, the following experiment was performed: The PSO is applied on all CEC 2005 benchmarks [15] which are scalable with respect to the search space dimensionality (some problems include matrices which are only given for at most 50 dimensions): f1, f2, f5, f6, f9, f12, f13, f15. Standard settings for the PSO as presented in Section 3 were used, except for that particles are not included in their own neighborhood so that for all particles $\boldsymbol{x}_{i,0} \neq \boldsymbol{l}_{i,0}$ holds. For each benchmark, 10,000 runs with 50 particles were performed.

The theoretical results were obtained by using Eq. (3) and Eq. (4).

	Theor. result	Conf. Exp.	PSO Exp.
$p'_A(1.496172, 0.72984, 1)$	0.17074	0.17072	0.149587
$p'_A(1.496172, 0.72984, 30)$	0.99636	0.99648	0.99582
$p'_A(1.496172, 0.72984, 100)$	0.9999999926	1	1

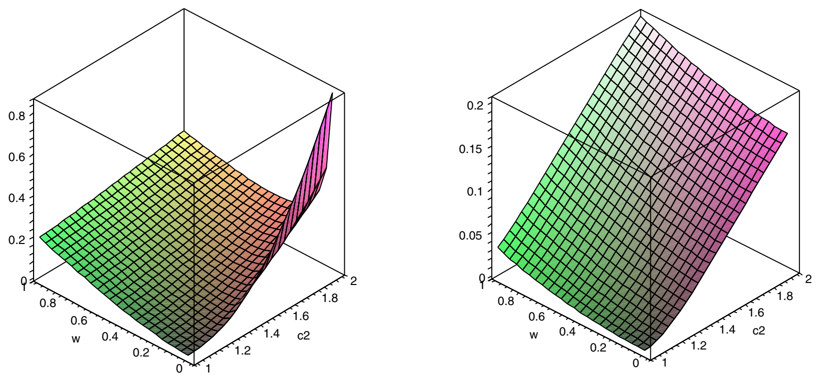


Fig. 1. The probabilities $p_A(c_2, \omega)$ (left) and $p_D(c_2, \omega)$ (right)

The comparison of the theoretical results and *Conf. Exp.* confirms the mathematical analysis. *PSO Exp.* shows that the assumption that local guides $\boldsymbol{l}_{i,0}$ are uniformly distributed at random in S is not too restrictive (see also subsequent examples) for higher-dimensional spaces. Moreover, the example demonstrates that the probability that a particle leaves the search space rapidly approaches 1 when increasing the search space dimensionality.

The probability $p_A(c_2, \omega)$ that a particle which is not its own local guide violates a specific boundary in the first iteration is shown in Fig. 1. It approaches zero for $c_2 \to 1$ and $\omega \to 0$. However, choosing such small values for c_2 and ω prevents exploration, and can therefore not be recommended.

Theorem 2. *If velocities are initialized with uniform distribution in $[-r, r]^n$, each particle i with $\boldsymbol{x}_{i,0} = \boldsymbol{l}_{i,0}$ leaves the n-dimensional search space in on average $\frac{\omega}{4} \cdot n$ dimensions in the first iteration.*

Proof. As $\boldsymbol{p}_{i,0} = \boldsymbol{l}_{i,0} = \boldsymbol{x}_{i,0}$, particle i's position $\boldsymbol{x}_{i,1}$ and velocity $\boldsymbol{v}_{i,1}$ in the first iteration evaluate to $\boldsymbol{v}_{i,1} = \omega \cdot \boldsymbol{v}_{i,0}$ and $\boldsymbol{x}_{i,1} = \boldsymbol{x}_{i,0} + \boldsymbol{v}_{i,1} = \boldsymbol{x}_{i,0} + \omega \cdot \boldsymbol{v}_{i,0}$. Hence, the d-th component of $\boldsymbol{x}_{i,1}$, $x_{i,1,d}$, is the sum of two independent, uniformly distributed random variables. Its density function $f_{x_{i,1,d}}$ is trapezoidal and can be determined by convolution:

$$f_{x_{i,1,d}}(z) = \begin{cases} \frac{1}{4\omega r^2} z + \frac{1}{4r} + \frac{1}{4\omega r} & \text{for } -r - \omega r < z \leq \omega r - r \\ \frac{1}{2r} & \text{for } \omega r - r < z < r - \omega r \\ -\frac{1}{4\omega r^2} z + \frac{1}{4r} + \frac{1}{4\omega r} & \text{for } -\omega r + r \leq z < r + \omega r \\ 0 & \text{otherwise} \end{cases}$$

Thus, the probability $p_B(\omega)$ that particle i exceeds the search space boundary in dimension d is $p_B(\omega) = \int_{-r-\omega r}^{-r} f_{x_{i,1,d}}(z)\mathrm{d}z + \int_{r}^{r+\omega r} f_{x_{i,1,d}}(z)\mathrm{d}z = \frac{\omega}{4}$. □

Corollary 1. *Each particle satisfying the assumptions of Theorem 2 leaves the n-dimensional search space in the first iteration, w.o.p.*

Proof. The probability $p'_B(\omega, n)$ that a particle which satisfies the given assumptions leaves the search space evalutes to

$$p'_B(\omega, n) = 1-(1-\omega/4)^n = 1-e^{-\Theta(n)}. \qquad \square$$

Example 2. We evaluate $p'_B(0.72984, 30) = 0.99763$, $p'_B(0.72984, 100) = 0.99999$, which shows that $p'_B(\omega, n)$ rapidly approaches 1 when increasing n.

From Theorem 1 and Corollary 1 follows that, w.o.p., all particles leave the search space in the first iteration, when initializing velocities uniformly at random in S.

2.2 Zero and Half-Diff Velocity Initialization

We will now show that using zero or half-diff initialization cannot avoid that many particles leave the search space in the first iteration, either.

Theorem 3. *If velocities are initialized to zero, each particle i with $\boldsymbol{x}_{i,0} \neq \boldsymbol{l}_{i,0}$ leaves the search space in the first iteration, w.o.p.*

Proof. Let particle i be an arbitrary particle satisfying the above condition. Its position and velocity in the first iteration are given by

$$\begin{aligned}\boldsymbol{v}_{i,1} &= \omega \cdot \boldsymbol{v}_{i,0} + c_1 \cdot \boldsymbol{r}_1 \odot (\boldsymbol{p}_{i,0} - \boldsymbol{x}_{i,0}) + c_2 \cdot \boldsymbol{r}_2 \odot (\boldsymbol{l}_{i,0} - \boldsymbol{x}_{i,0}) = \\ &= c_2 \cdot \boldsymbol{r}_2 \odot (\boldsymbol{l}_{i,0} - \boldsymbol{x}_{i,0}) \\ \boldsymbol{x}_{i,1} &= \boldsymbol{x}_{i,0} + \boldsymbol{v}_{i,1} = (\boldsymbol{1} - c_2 \cdot \boldsymbol{r}_2) \odot \boldsymbol{x}_{i,0} + c_2 \cdot \boldsymbol{r}_2 \odot \boldsymbol{l}_{i,0} \ .\end{aligned}$$

For fixed $\boldsymbol{r}_2$, $x_{i,1,d}$ is the sum of two non-identical uniformly distributed random variables, and therefore trapezoidally distributed. If $r_{2,d} < \frac{1}{c_2}$, particle i does not violate the boundary in dimension d. Otherwise, the density function $f_{x_{i,1,d}}$ of $x_{i,1,d}$ can be computed (omitted due to space constraints), and the probability $q_2(r_{2,d}, c_2)$ that particle i crosses the search space boundary in dimension d is

$$q_2(r_{2,d}, c_2) = \begin{cases} \int_{-\infty}^{-r} f_{x_{i,1,d}}(z)\mathrm{d}z + \int_{r}^{\infty} f_{x_{i,1,d}}(z)\mathrm{d}z = 1 - \frac{1}{c_2 r_{2,d}} & \text{if } r_{2,d} > \frac{1}{c_2} \\ 0 & \text{otherwise} \end{cases}$$

As $r_{2,d}$ is uniformly distributed between 0 and 1, we finally determine the probability $p_C(c_2)$ that a particle violates the boundary in a specific dimension to

$$p_C(c_2) = \int_0^1 q_2(r_{2,d}, c_2)\mathrm{d}r_{2,d} = \frac{-1 - \ln(c_2) + c_2}{c_2} \ . \tag{5}$$

From $c_2 > 1$, $p_C > 0$ follows. Thus, the probability $p'_C(c_2)$ that particle i leaves the n-dimensional search space evaluates to

$$p'_C(c_2, n) = 1 - (1 - p_C(c_2))^n = 1 - e^{-\Theta(n)} \ . \tag{6}$$

$\square$

Example 3. For the theoretical results, Eq. (5) and Eq. (6) were used.

	Theor. result	Conf. Exp.	PSO Exp.
$p'_C(1.496172, 1)$	0.06233	0.06223	0.04548
$p'_C(1.496172, 30)$	0.85497	0.85574	0.83358
$p'_C(1.496172, 100)$	0.988	0.998	0.99801

The example confirms the theoretically derived formulas for $p_C(c_2)$ and $p'_C(c_2, n)$, and the relevance of the theoretical results for high-dimensional PSO application.

Conjecture 1. If the particles' velocities are initialized according to the *half-diff* strategy, each particle i with $\boldsymbol{x}_{i,0} \neq \boldsymbol{l}_{i,0}$ leaves the search space, w.o.p.

Similar to the proof for Theorem 1, the probability $p_D(c_2, \omega)$ that a particle leaves the search space in dimension d can be evaluated to

$$p_D(c_2,\omega) = \int_0^1 \left(\int_{-\infty}^{-r} f_{x_{i,1,d}}(z)\mathrm{d}z + \int_r^{\infty} f_{x_{i,1,d}}(z)\mathrm{d}z \right) \mathrm{d}r_{2,d} =$$

$$\begin{aligned} &= (12\omega c_2(\omega-2))^{-1}\cdot(32\omega-22\omega^2+3\omega^3-16\ln(2)+24\omega\ln(2)+16\ln(2-\omega) \\ &\quad -24\omega\ln(2-\omega)+24\ln(c_2)\omega-12\omega^2\ln(2)+12\omega^2\ln(2-\omega)-12\ln(c_2)\omega^2+2\omega^3\ln(2) \\ &\quad -2\omega^3\ln(2-\omega)+2\ln(c_2)\omega^3+2\omega^3\ln(\omega)-24\omega c_2+12\omega^2 c_2-2\omega^3\ln(\omega-2+2c_2)) \end{aligned} \tag{7}$$

and is plotted in Fig. 1 (right). The probability that a particle leaves the n-dimensional search space in the first iteration is

$$p'_D(c_2, \omega, n) = 1 - (1 - p_D(c_2, \omega))^n \tag{8}$$

which is overwhelming if $p_D(c_2, \omega) > 0$. Fig. 1 shows that there is strong evidence that $p_D(c_2, \omega) > 0$, at least for commonly used values for c_2 and ω, e.g., $c_2 > 1.1$ and $\omega > 0.3$.

Example 4. For the theoretical results, Eq. (7) and Eq. (8) were used. Again, the theoretical results are confirmed:

	Theor. result	Conf. Exp.	PSO Exp.
$p'_D(1.496172, 0.72984, 1)$	0.094572	0.094661	0.077998
$p'_D(1.496172, 0.72984, 30)$	0.949229	0.950656	0.941712
$p'_D(1.496172, 0.72984, 100)$	0.999952	0.999954	0.999935

2.3 Consequences for PSO Application

The theoretical analysis showed that none of the three investigated velocity initialization strategies can avoid that many particles leave high-dimensional search spaces as early as in the first iteration. Even initializing velocities to zero cannot prevent particle explosion. For PSO application, there exist several strategies to deal with this observation:

In order to avoid that particles leave the search space, bound handling strategies which keep the particles inside the search space, such as *Absorb* or *Random*, can be applied. Many other strategies exist in the literature [2,3,4,5]. From the methods studied in the experimental analysis in Section 3, *Absorb* performed best. However, hybrid methods as proposed by Clerc [2] seem to be promising.

When using *Infinity* bound handling, which is a commonly used strategy [6], many particles mainly explore invalid space at the beginning of the optimization process. In our experiments (see Section 3), *Infinity* was significantly outperformed by *Random* and *Absorb* on almost all 100-dimensional problems. Velocity clamping can help particles to reenter the search space.

More general approaches for constraint handling, e.g., the use of penalty functions, can also be applied to deal with the initial particle explosion.

3 Experimental Results

The following experimental analysis studies the impact of velocity initialization and bound handling in particle swarm optimization. For all experiments, the PSO with standard parameter settings [6] (50 particles, $c_1 = c_2 = 1.496172, \omega = 0.72984$) was applied on four widely-used benchmarks, Sphere, Rosenbrock, Rastrigin, and Griewank (function descriptions and particle initialization ranges see, e.g., [6]), and on the CEC 2005 benchmarks f1, f2, f5, f6, f9, f12, f13, f15 [15]. When solving a CEC benchmark, particles were initialized uniformly at random in S. The swarm is connected via the *von Neumann* topology, a two-dimensional grid with wrap-around edges [16]. A particle is included in its own neighborhood.

The PSO terminated after $100,000$ function evaluations, and each configuration was repeated 100 times. In order to compare the performance of two algorithms A and B, the one-sided Wilcoxon rank sum test was used with null-hypothesis $H_0 : F_A(z) = F_B(z)$, and the one-sided alternative $H_1 : F_A(z) < F_B(z)$ (where $F_X(z)$ is the distribution of the results of algorithm X). Statistical significance was evaluated on a significance level of 0.01.

3.1 Velocity Initialization

In the theoretical analysis, three different velocity initialization strategies were studied. Uniform initialization causes all particles to leave the search space, w.o.p, whereas zero initialization slows down initial exploration. Hence, although none of the strategies can avoid that many particles leave high-dimensional search spaces, w.o.p., half-diff initialization seems to have fewest drawbacks.

In order to confirm this assumption, experiments were conducted on all 100-dimensional benchmarks. The following tables summarize the one-sided Wilcoxon rank sum test. For each algorithmic combination (A, B) the matrices show how often A significantly outperformed B. E.g., entry "1" in the first table shows that uniform significantly outperformed half-diff on one benchmark (out of 12).

Absorb bound handling

	1	2	3
uniform (1)	0	2	1
zero (2)	2	0	0
half-diff (3)	4	4	0

Random bound handling

	1	2	3
uniform (1)	0	0	0
zero (2)	0	0	0
half-diff (3)	2	2	0

Table 1. Comparison of bound handling strategies. The tables show summaries of one-sided Wilcoxon rank sum tests with significance level 0.01. $\mathcal{S}$ is the set of all benchmarks. *Example:* Entry {f5, f9} in the first table shows that *Absorb* performed significantly better than *Infinity* on f5 and f9.

Half-diff velocity initialization, 30 dimensions

	Absorb	Infinity	Random
Absorb	–	{f5, f9}	{f1, f2, f5, f6, f9, f12, f15}
Infinity	{f2, f12, f15}	–	{f1, f2, f5, f6, f12, f15}
Random	{Rastrigin}	{Rastrigin, Griewank}	–

Half-diff velocity initialization, 100 dimensions

	Absorb	Infinity	Random
Absorb	–	$\mathcal{S}\backslash${Sph., Rosenbr.}	{f1, f2, f5, f6, f9, f12, f15}
Infinity	–	–	–
Random	{Sph., Rosenbr., Rastr.}	$\mathcal{S}\backslash${Sph.}	–

Half-diff velocity initialization provides slightly better results than the other two strategies, and can therefore be recommended for PSO application.

3.2 Bound Handling

The theoretical analysis showed that many particles leave the search space at the beginning of the optimization process. To each of these particles, the bound handling procedure is applied, and therefore, bound handling strongly influences the particle swarm behavior, at least in the early steps of the algorithm. In order to check experimentally whether the bound handling method actually strongly influences particle swarm performance and whether the effect is stronger when more search space dimensions are involved, three bound handling strategies (*Absorb*, *Random*, and *Infinity*) were investigated on 30- and 100-dimensional optimization problems. Half-diff velocity initialization was used. The results are shown in Table 1, and confirm significant performance differences, especially for the 100-dimensional benchmarks.

4 Conclusion

Particle trajectories during the first iteration were investigated theoretically for three widely-used velocity initialization strategies. It was proven that many particles leave the search space as early as in the first iteration. To be more precise: Uniform velocity initialization causes all particles to leave the search space with overwhelming probability (w.o.p.) with respect to the search space dimensionality n. In order to reduce the number of particles leaving the search space, other velocity initialization strategies, among them zero [13] and half-diff [11] initialization, were proposed. However, this study showed that still many particles leave the search space, w.o.p. Examples demonstrated that this probability rapidly approaches 1 if the search space dimensionality is increased. Consequences for PSO application and strategies to deal with this observation were derived.

The presented analysis highlights the importance of bound handling for PSO: As the bound handling procedure is applied to many particles at the beginning of the optimization process, it strongly influences particle swarm behavior. The experimental study confirms significant performance differences when varying the bound handling method.

References

1. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
2. Clerc, M.: Confinements and Biases in Particle Swarm Optimization (2006), `http://clerc.maurice.free.fr/pso/`
3. Alvarez-Benitez, J.E., Everson, R.M., Fieldsend, J.E.: A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In: Evolutionlary Multi-Criterion Optimization, pp. 459–473 (2005)
4. Zhang, W.J., Xie, X.F., Bi, D.C.: Handling Boundary Constraints for Numerical Optimization by Particle Swarm Flying in Periodic Search Space. In: Proceedings of the IEEE Congress on Evol. Computation, vol. 2, pp. 2307–2311 (2004)
5. Robinson, J., Rahmat-Samii, Y.: Particle Swarm Optimization in Electromagnetics. IEEE Transactions on Antennas and Propagation 52(2), 397–407 (2004)
6. Bratton, D., Kennedy, J.: Defining a Standard for Particle Swarm Optimization. In: Proc. IEEE Swarm Intelligence Symp., pp. 120–127 (2007)
7. Clerc, M.: Particle Swarm Optimization. ISTE Ltd (2006)
8. Clerc, M., Kennedy, J.: The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
9. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317–325 (2003)
10. Jiang, M., Luo, Y.P., Yang, S.Y.: Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. Information Processing Letters 102(1), 8–16 (2007)
11. Clerc, M., et al.: Standard PSO 2007 (2007), `http://www.particleswarm.info` (standard_pso_2007.c)
12. Helwig, S., Wanka, R.: Particle Swarm Optimization in High-Dimensional Bounded Search Spaces. In: Proc. IEEE Swarm Intelligence Symp., pp. 198–205 (2007)
13. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. John Wiley and Sons Ltd, Chichester (2005)
14. Bradley, D., Gupta, R.: On the Distribution of the Sum of n Non-Identically Distributed Uniform Random Variables. Annals of the Institute of Statistical Mathematics 54(3), 689–700 (2002)
15. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. KanGAL Report 2005005, Nanyang Technological University, Singapore (2005)
16. Kennedy, J., Mendes, R.: Population Structure and Particle Swarm Performance. In: Proceedings of the IEEE Congress on Evol. Computation, pp. 1671–1676 (2002)

Large-Scale Optimization of Non-separable Building-Block Problems

David Iclănzan and D. Dumitrescu

Babeş-Bolyai University,
Department of Computer Science,
400084 Kogălniceanu no. 1, Cluj-Napoca, Romania
david.iclanzan@gmail.com,
ddumitr@cs.ubbcluj.ro

Abstract. This paper presents principled results demonstrating how the identification and exploitation of variable dependencies by means of Artificial Neural Network powered online model building, combined with a model based local-search, opens the way towards large-scale optimization of hard, non-separable building-block problems.

Keywords: large-scale optimization, online model building, model based local-search, adaptive neighborhood structure, scalability analysis.

1 Introduction

Challenging optimization problems with staggering number of variables require the development of techniques able to efficiently and reliably address, difficult, large-scale problems.

The feasible sizes of problems strongly depend on their particular structure. Recent results had shown that with an efficient parallel implementation, competent algorithms can solve *separable* problems with millions to billions of variables even in the presence of added noise [1]. However, for solving effectively and efficiently *non-separable*, difficult, large-scale problems, where identification and exchange of higher order building-blocks is mandatory, more powerful, context aware methods are required, capable of performing a proper problem decomposition.

Probabilistic model building methods or Estimation of Distribution Algorithms (EDAs) can render larger problems feasible by identifying and exploiting dependencies. Nevertheless, they require populations large enough to guarantee proper initial-supply, decision-making and accurate model-building.

The population requirements and model building costs are problematic for large problem instances. Following known population sizing theories [2,3] results that the solving of modular, non-separable problems up to millions of variables would require terabytes of memory to accommodate the necessary population. In addition, the search for an appropriate model in EDAs capable of modeling higher order dependencies, requires many model evaluations with regard to the population. Given the implied population sizes as the dimension of the problems increases, the computational cost of model building quickly exceeds economical practicality.

In order to solve non-separable problems up to millions of variables, we need a method that is computationally efficient in terms of model building and also very efficacious in terms of memory usage.

For meeting these desiderates, we consider the extension of the model based local-search presented in [4] with an online learning and model building mechanism. The proposed Online Model Based Local-Search (OMBLS) framework employs an adaptive neighborhood structure which facilitates the operation directly on modules. Nevertheless, this approach does not use a memory to store semi-converged solutions for later analysis, one point is sampled at the time and the search experience is accumulated and information about the problem structure is inferred from a single data structure, resulting in very low memory requirements.

The great advantage of the online learning, in addition to the low memory requirement, is that the learning is automatic. Using vector quantization, probability density functions are modeled by the distribution of prototype vectors; there is no model search and repeated costly evaluation against a set of samples like in population based EDAs.

2 The Mixed Hierarchical Test Function

To obtain a single, large, scalable test problem which embeds all the test features found in separate suites, we consider the mixing of three standard and well known hierarchical test functions: the hierarchical IFF [5], the hierarchical XOR [6] and the hierarchical trap function [7].

These problems are defined on binary strings of the form $x \in \{0,1\}^{k^p}$, where k is the number of sub-blocks in a block, and p is the number of hierarchical levels.

As $k = 2$ for hIFF and hXOR, respectively $k = 3$ for hTrap in this paper, a Mixed Hierarchical function (hMix) with p hierarchical levels will have $n = 2^p + 2^p + 3^p = 2^{p+1} + 3^p$ number of variables. The n problem variables are shuffled by random reordering; the original arrangement (un-shuffling) for a variable x is given by a decode function $\chi(x)$. The decoded state is partitioned in three subsets, one for each of the three components of the problem by $\chi_1(x), \chi_2(x), \chi_3(x)$, where $\chi_1(x)$ covers the first 2^p variables from $\chi(x)$, $\chi_2(x)$ the following 2^p ones, whilst $\chi_3(x)$ retrieves the last 3^p ones.

The hMix with p hierarchical levels is then given by:

$$hMIX_p(x) = hIFF(\chi_1(x)) + hXOR(\chi_2(x)) + hTrap(\chi_3(x)) \qquad (1)$$

As two out of the three components have two global optima (hIFF and hXOR), the combined hMix will have four globally optimal states.

Although having a gross-scale building-block structure, hMix and its components are hard to solve without proper problem decomposition as the blocks are not separable. Context aware methods must be applied which can overcome the conceptual separation between the fitness and meaning (context dependence) of building-blocks.

Being formed from multiple components, the computation of hMix is eased. Local-search techniques usually alter one or just few variables, perturbing only a few blocks of a single component. Thus, efficacious caching can be employed for this particular case; the values of the other components and unperturbed modules can be used from previous epochs without the need for recomputing.

3 Online Model Based Local-Search

In Model Based Local-Search [8,4] the changing of the fixed problem representation with a problem structure aware description, leads to a very efficient search and can even lead to the solving of hierarchical problems by repeated decomposition [4].

Following the framework proposed in [4], OMBLS has two main phases. The first one refers to the accumulation of search experience, provided by the repeated model based local-search. The second phase concerns the exploitation of search experience by linkage learning and module knowledge update, followed by the collapse of the search space.

3.1 Employed Local-Search

The locally optimal setting of modules in hMix are in immediate vicinity of randomly initialized module configurations, as module sizes are less or equal than three and each module has two context optimal settings. Accordingly, we use a simple greedy search among these configurations, alike the method used in [4].

For problems with larger modules, where context-optimal settings are harder to reach, a more powerful local-search method may be required. A model based local-search, based on macro-mutation [9] search strategy, had shown the ability to identify and exploit much larger module sizes [10].

The building-block hill-climbing employed in this paper is rather straightforward: each module is processed systematically by testing its configurations and selecting the one which provides the best objective function value. While the search for the optimal configuration of a particular block is carried out, the configurations of the other modules are held still.

3.2 Linkage Learning within OMBLS

When applying the greedy search on the presented hierarchical problems, the method will always discover a context-optimal setting for each module, but due to the non-linear interdependencies between the modules organized hierarchically, it will converge to a local optimum in most of the cases.

As there are only two context-optimal settings for each module, the variables of the converged solutions will be grouped in a subspace which has lower dimensionality than the dimensionality of the data. Certain artificial neural network models have the ability to adaptively process input patterns and to produce simpler patterns with fewer components in accordance with the topology of the input space. On longer term, by training the network with multiple solutions, the modular structure of the problem can be inferred.

A network which seeks to preserve the topological properties of the input space is the Self Organizing Map (SOM) [11]. The network is trained using *unsupervised learning* to produce a two dimensional, discretized representation of the input space, called a map.

The SOM weight adapting algorithm is based on the competitive learning paradigm; vector quantization is used to model probability density functions by the distribution of prototype vectors. Concretely, when a sample s is presented to the network, the following steps are executed:

1. The Euclidean distance to all weight vectors is computed.
2. The neuron with weight vector most similar to the input is nominated as the best matching unit (BMU).
3. The weights of the BMU and neurons in the neighborhood are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU.

This algorithm can be very well iteratively updated online with "live" data, directly inputting the results (locally converged states) of the model based local-search, rather than training with samples from a memory.

In the proposed method we use a SOM with a lattice of $5 * 5$ neurons, which are arranged in a rectangular grid with regular spacing. A weight vector of size n where n is the number of detected modules and a position in the map space is associated with each neuron. All weights are initialized with the value 0.5. To induce a symmetric negative bias into the adjustment of the weights, the 0's from the inputs are replaced with -1.

The network can be trained with data for an apriori settled number of epochs or one can use a dynamic stopping condition, which checks if the change in the last few epochs of the weights values is below a certain threshold.

After training the SOM online, dependencies are deduced form the internal representation of the network based on the heuristic that similar inputs should produce similar patterns in their associated weights i.e dependent inputs have roughly the same values for their weights.

Accordingly, we use the following metric for detecting the dependency between variables x_i and x_j:

$$d(x_i, x_j) = \sum_{l=1}^{r} ||W_{il}| - |W_{jl}|| \tag{2}$$

where r is the number of neurons on the rectangular lattice and W represents the weights of the SOM. This relation measures the closeness of different variables by taking into account the weights related to them.

We consider x_i and x_j as being dependent if $d(x_i, x_j) \leq \epsilon$ for a predefined ϵ threshold.

Dependent variables will be merged into the same composite module for the next phase of the search.

By analyzing the weights of the network, we are able to decide which of the current variables are linked, but in order to collapse the search space we also need their context-optimal settings. As a consequence, provided with only the variable relationships, for each new composite block the method must search all the possible combinations of sub-modules in the context of randomly generated states and retain the best λ ones.

3.3 Collapsing the Search Space

For efficient implementation, where module knowledge can be naturally represented in binary mode, in this paper we only focus on the case where $\lambda = 2$, although the method is not limited to this special case.

The two most fit schemata found by the exhaustive search are compressed into one single bit, where 1 maps to the most expressive schema and 0 maps to the second most expressive one. The information about the schemata expressed by each bit is recorded in a map M. The other $2^k - 2$ configurations are discarded for the remainder of the search leading to the collapse of the search space.

The following section summarizes the Online Model Based Local-Search.

3.4 OMBLS Algorithm

Starting from a representation in concordance with the original problem, in a first phase, search experience is accumulated by training the SOM online with the "live" states provided by repeated local-search working on the current representation, which always express the most two fit schemata found at a lower level. The local-search strategy used must be powerful enough to discover fully optimized modules at a single hierarchical level.

After convergence of the network, in the second phase the structure of the input space is inferred from the weights of the network and expressed by variable linkages. Furthermore, an exhaustive search is performed according to the detected linkages, to find the best context-optimal settings for each new module.

The search space is collapsed as the module aware representation is compressed according to the detected linkages and their two most fit context-optimal settings.

After these steps, the search enters again phase one, with the local-search operating on the newly derived representation, further exploring the combinative neighborhood of the detected modules.

The method stops when the search space cannot be collapsed anymore or a predefined number of objective function evaluations is exceeded.

OMBLS can conquer hierarchical difficulty and is able to overcome the non-linear interdependence between building-blocks, as it applies a proper decomposition at each level, promising, competing sub-solutions are kept and the representation is adapted by expressing detected modules at lower level as variables of the upper level.

Formally the OMBLS is outlined in Algorithm 1.

Algorithm 1. Online Model Based Local-Search

```
Data: $n, n_S, \epsilon$,@stopping_cond.
/* Initially each binary variable is a base module                    */
for $i = 1, n$ do
    $M[i][0] \leftarrow \{0\}$;
    $M[i][1] \leftarrow \{1\}$;
while not @stopping_cond do
    /* PHASE I                                                         */
    /* Build the SOM                                                   */
    $net \leftarrow InitializeSOM(n)$;
    for $i = 1, n_S$ do
        /* Generate a random binary state of length $n$                */
        $s \leftarrow RandomState(n)$;
        /* Apply module-wise greedy search                             */
        $s \leftarrow GreedySearch(s, M)$;
        /* Train the network online using vector quantization          */
        $net \leftarrow Train(net, s)$;
    /* PHASE II                                                        */
    /* Detect possible modules via weight analysis                     */
    $nm \leftarrow GetLinkages(net.Weights, \epsilon)$;
    /* Identify the two most fit schemata for new modules by
       exhaustive search                                               */
    $CO_{set} \leftarrow RetrieveBestTwoSettings(nm)$;
    /* Collapse the search space and update the building-block
       configuration according to the detected modules and their
       context-optimal settings                                        */
    $n \leftarrow |CO_{set}|$;
    for $i = 1, n$ do
        if module $i$ is new then
            $M[i][0] \leftarrow CO_{set}[i][0]$;
            $M[i][1] \leftarrow CO_{set}[i][1]$;
```

4 Run-Time and Scaling of the OMBLS

Assuming that the used machine learning technique successfully detects the correct dependencies, the global convergence of the OMBLS on the studied test suite can be proven, by showing that there is a path towards global optima on each component, easily followed by the method.

As mentioned in Sec. 3.1, in the components of hMix, the number of sub-blocks in a block at each hierarchical level is maximally three and there are two context optimal settings for each module. If the partial knowledge about the problem structure is correct, starting from a random combination of sub-blocks, if the global optima have not yet been attained, one of the two context-optimal setting at the next level is always reached by the greedy search[1]. As fully-optimized modules are found they are expressed as variables in the next phase of the search, which leads to a simple recursive solving of all hierarchical levels as in [4].

Provided that the SOM indicates the correct dependencies, the correct partial knowledge about the problem structure is guaranteed by the fact that the method performs an exhaustive search in order to determine the two most suitable settings for each module.

If the applied local-search strategy is able to find the correct context-optimal settings at each hierarchical level (true for the greedy search on hMix), the method will follow the above mentioned hierarchical-recursive path and thus, a tight upper bound on the performance of OMBLS can be given.

In the case of the module-wise greedy search, at one hierarchical level, the number of objective function evaluations is in concordance with the number of modules l, the number of context-optimal setting $\lambda = 2$ for each module and the number of epochs used to feed the network n_S:

$$T_{LS} = 2 \cdot n_S \cdot l \tag{3}$$

The search for the context-optimal settings will take a number of objective function evaluations exponential in the size (denoted by k_i) of the newly discovered modules M' and the number of context-optimal settings:

$$T_{COS} = \sum_{i=1}^{|M'|} 2^{k_i} \tag{4}$$

For p hierarchical levels, the upper bound is given by the summation of the model based local-search T_{LS} and the search for context-optimal settings T_{COS} on each hierarchical level.

As we know that the module sizes on hIFF and hXOR equal $k_1 = 2$ and for hTrap the module size is $k_2 = 3$, we got the following upper bound on hMix for p hierarchical levels:

$$T_{hMix_p} = 2 \cdot \sum_{\substack{l=k_1 \\ l=l*k_1}}^{k_1^p} \left(2 \cdot n_S \cdot l + \frac{l}{k_1} \cdot 2^{k_1}\right) + \sum_{\substack{l=k_2 \\ l=l*k_2}}^{k_2^p} \left(2 \cdot n_S \cdot l + \frac{l}{k_2} \cdot 2^{k_2}\right) \tag{5}$$

To empirically confirm this result and to test the efficiency of the SOM based online linkage detection technique, the scalability of the OMBLS have been tested on hMix with $p = \{4, 6, 8, 10, 11\}$ hierarchical levels, with the resulting problem sizes $n = \{113, 857, 7073, 61097, 181243\}$. Theory predicts [2], that the solving of hMix with $p = 11$ by classic EDAs using random sampling, requires a population with more than 10 million members as population size scales as $N = 2^{2k}m \cdot$

[1] The proof, omitted here for space consideration, is based on showing that the Hamming distance between a context-optimal setting at the next level, and its randomly initialized component modules at the lower level, is at most 1.

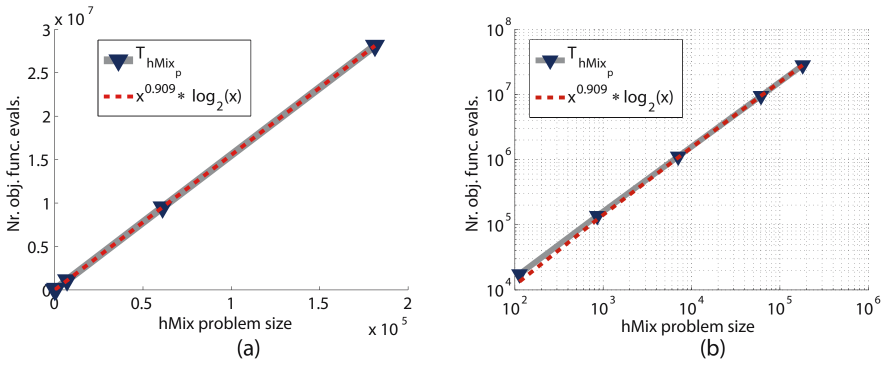


Fig. 1. Sub-linearithmic scaling of OMBLS with module-wise greedy local-search strategy on hMix: (a) arithmetic plot; (b) logarithmic plot

$log(m)$, where m is the number of modules. With such a huge number of samples, the cost of model building an extraction from the population may be of order of quadrillions in a single generation.

Due to the deterministic nature of the algorithm, with precisely predictable behavior, for $p \leq 8$ a total number of 25 independent runs were averaged while for $p = \{10, 11\}$ the number of runs was restricted to 10.

As the input samples of the network are high-quality, converged states, we restricted the number of module-wise local-search epochs to $n_S = 50$. The threshold for linkage detection was set to $\epsilon = 1.0e - 6$.

The method found one of the four global optima in all cases confirming the efficiency of the online SOM based linkage learning. The scaling of the method on hMix is presented in Fig. 1. The experimental result was approximated with a function of the form $f(x) = ax^b \cdot log_2(x)$ where a and b are determined by the least square error method. As depicted, OMBLS scales on hMix approximately as $\theta(x^{0.909} \cdot log_2(x))$, where x is the problem size.

Potentially, the most costly operation of OMBLS is represented by the search for context-optimal settings, which is exponential in the order of dependencies revealed by the SOM based learning. This phenomenon is not particular for our method. The size of population in EDAs (implicitly the number of objective function evaluations in each generation) is also lower bounded by the exponential of the order of dependencies covered by the probabilistic model.

Nevertheless, for boundedly-difficult problems like the hMix, where the order of dependencies is low compared to the problem size ($k << n$), the computational cost of the search for context-optimal settings can be approximated with a constant. Consequently, the upper bound on the objective function evaluations is dominated by the computational complexity of the employed local-search, being $log_2(n) \cdot O(T_{LS})$, as we can have a maximum of $log_2(n)$ hierarchical levels.

In our case, as the cost of the greedy search is linear in the number of modules, from Eq. (5) results a very efficient sub-linearithmic running time, confirmed empirically by our experiments.

If we used a model based (1+1)EA instead of greedy search, which provably converges on hMix to context-optimal settings at each hierarchical level in $O(n \cdot log_2(n))$, the complexity of OMBLS would become $O(n \cdot log_2(n)^2)$.

The memory requirement of the OMBLS is very low, being linear in the problem size.

The running time of model building is quadratic in the number of modules as the distance according to Eq. (2) must be computed for each pairwise combination of modules. Even if n^2 running time is mostly considered as competent, it can become problematic when n is of order of millions. To efficiently address in wall-clock time such large inputs, a parallel model building must be employed; parallelization is a feasible and easily achievable desiderate (given the proper hardware) as distance measurements of different module pairs can be computed independently.

5 Conclusions and Further Work

The paper presents a model based trajectory framework, namely the Online Model Based Local-Search (OMBLS) that learns the problem structure online by means of topology preserving SOMs. OMBLS operates via hierarchical decomposition, detected modules are used to collapse the search space and reformulate the optimization problem with discovered modules and their context-optimal settings as new search variables.

For boundedly-difficult, non-overlapping, non-separable building-block problems, the cost of the OMBLS is upper bounded by the number of hierarchical levels, multiplied with the running complexity required by the module-wise local-search to converge to context-optimal settings at one hierarchical level.

As training is done online, the memory requirements of the method are limited to storing one solution at the time, the two most fit schemata for each module and a SOM which is also linear in the number of input variables. Nevertheless, the network can only reveal variable dependencies; a further search for context-optimal settings must be employed, resulting in a model building complexity in terms of objective function evaluations, bounded by the exponential of the order of dependencies detected by the online learning. This computational cost is similar with the population requirement of classic EDAs, where the number of samples is also lower bounded by the exponential of the order of dependencies covered by the probabilistic model.

OMBLS opens the way towards large-scale optimization of hard, non-separable building-block problems. Further work will focus on probabilistic recoding of the information ("soft" chunking) in order to tackle problems with more complicated structure. Another line of research will focus on the parallelization of the proposed framework: instead of sequential epochs, several model based local-searches can be run concomitantly on different computational units; model building can be also greatly parallelized as the search for context-optimal settings for each building-block, respectively the pairwise distance computation of different modules can be run in parallel.

Acknowledgments

This work was sponsored by MEdC-CNCSIS and by the Sapientia Institute for Research Programs (KPI).

References

1. Sastry, K., Goldberg, D.E., Llora, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 577–584. ACM, New York (2007)
2. Yu, T.L., Sastry, K., Goldberg, D.E., Pelikan, M.: Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In: Lipson, H. (ed.) GECCO, pp. 601–608. ACM, New York (2007)
3. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: Bayesian optimization algorithm, population sizing, and time to convergence. In: Whitley, L.D., Goldberg, D.E., Cantú-Paz, E., Spector, L., Parmee, I.C., Beyer, H.G. (eds.) GECCO, pp. 275–282. Morgan Kaufmann, San Francisco (2000)
4. Iclanzan, D., Dumitrescu, D.: Overcoming hierarchical difficulty by hill-climbing the building block structure. In: Thierens, D., et al. (eds.) GECCO 2007: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation, vol. 2, pp. 1256–1263. ACM Press, London (2007)
5. Watson, R.A., Hornby, G., Pollack, J.B.: Modeling building-block interdependency. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V 1998. LNCS, vol. 1498, pp. 97–108. Springer, Heidelberg (1998)
6. Watson, R.A., Pollack, J.B.: Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In: Brave, S. (ed.) GECCO 1999: Late Breaking Papers, Orlando, Florida, USA, July 13, 1999, pp. 292–297 (1999)
7. Pelikan, M., Goldberg, D.E.: Escaping hierarchical traps with competent genetic algorithms. In: Spectes, L., et al. (eds.) GECCO 2001, pp. 511–518. Morgan Kaufmann, San Francisco (2001)
8. Sastry, K., Goldberg, D.E.: Designing competent mutation operators via probabilistic model building of neighborhoods. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 114–125. Springer, Heidelberg (2004)
9. Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, University of New Mexico, Albuquerque, NM (1995)
10. Iclanzan, D., Dumitrescu, D.: Going for the big fishes: Discovering and combining large neutral and massively multimodal building-blocks with model based macro-mutation. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and Evolutionary Computation. ACM Press, New York (accepted, 2008)
11. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics 43, 59–69 (1982)

Particle Filter with Swarm Move for Optimization

Chunlin Ji[1], Yangyang Zhang[2], Mengmeng Tong[3], and Shengxiang Yang[4]

[1] Department of Statistical Sciences, Duke University
Durham, North Carolina 27708, USA
[2] Department of Engineering Science, University of Oxford
Parks Road, Oxford OX1 3PJ, UK
[3] School of Information Science and Technology
Northeastern University, Shenyang 110004, China
[4] Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, UK

Abstract. We propose a novel generalized algorithmic framework to utilize particle filter for optimization incorporated with the swarm move method in particle swarm optimization (PSO). In this way, the PSO update equation is treated as the system dynamic in the state space model, while the objective function in optimization problem is designed as the observation/measurement in the state space model. Particle filter method is then applied to track the dynamic movement of the particle swarm and therefore results in a novel stochastic optimization tool, where the ability of PSO in searching the optimal position can be embedded into the particle filter optimization method. Finally, simulation results show that the proposed novel approach has significant improvement in both convergence speed and final fitness in comparison with the PSO algorithm over a set of standard benchmark problems.

1 Introduction

Particle filter, also known as sequential Monte Carlo (SMC), is a class of importance sampling and resampling techniques designed to simulate from a sequence of probability distributions, which has gained popularity for the last decade to solve sequential Bayesian inference problems [1]. It was recently extended to a general framework to deal with static and sequential Bayesian inference, as well as the global optimization. In order to deal with an optimization problem, a sequence of artificial dynamic distribution was designed to employ the particle filter algorithm. The basic idea of particle filter optimization (PFO) method was first presented in our previous works [2] to solve discrete optimization problems in wireless communication system. The crucial element in the PFO algorithm is how to design the system dynamic function, which forces the set of particles to move toward the 'promising' area containing optima.

Particle Swarm Optimization (PSO) is a well studied heuristic optimization technique inspired by the social behavior observable in nature, such as flocks of birds and schools of fish [3]. In a basic PSO algorithm, a set of particles is generated randomly, and their positions (states) are iteratively updated according to their own experience and the experience of the swarm (or neighboring particles). The heuristic strategy of swarm move works well and therefore enable the PSO algorithm being effective and efficient. However, the PSO is not universal for all optimization problems because it suffers the premature convergence which enables the PSO algorithm easily to get stuck in local minima.

A novel algorithmic framework is presented in this paper, where the swarm move strategy is incorporated into particle filter optimization algorithm. The update equation of particle swarm move in PSO algorithm is treated as the system dynamic of a state space model, while the objective function in optimization problem is designed as the observation model to motivate the swarm moving toward the optimal position. In particle filter, the particles first evolve according to the system dynamic model where they 'learn' the information from the whole population. After that, the particles are updated by information from the observation model where they 'learn' the information from the objective function. Therefore, the particles move towards location of the global optima sequentially by 'learning' the information from these two aspects. By using the swarm move strategy as the system dynamic, the desirable searching mechanism of PSO is incorporated into the PFO algorithm and enhance its searching ability. The proposed novel algorithm incorporates the state space probability modelling and resample strategy to the PSO algorithm, which potentially enhance the ability of PSO algorithm in two aspects: making it easier to jump out local optima and further refining the final results.

This paper is organized as follows. Section 2 reviews the basic PSO algorithm. Section 3 introduces the basic particle filter algorithm. In Section 4, we propose the particle filter with particle swarm move for optimization problem. In section 5, the proposed algorithm is tested on a set of benchmark problems. Finally, we conclude the paper in Section 6.

2 Particle Swarm Optimization

2.1 The Basic PSO Algorithm

The basic PSO algorithm first starts with a number of particles which are randomly generated in the function domain space. After that, each particle flies through the search space with a velocity which is dynamically adjusted according to its own flying experience and the experience from neighboring particles. Specifically, the behavior of each particle is affected by either the local best or the global best particle to help it fly through a hyperspace. Therefore, by observing the behavior of the flock and memorizing their flying histories, all particles in the swarm can quickly converge to near-optimal geographical positions [3]. The particles are updated according to the following equations:

$$v_n = wv_{n-1} + \phi_1(x_{ibest} - x_{n-1}) + \phi_2(x_{gbest} - x_{n-1}) \tag{1}$$

$$x_n = x_{n-1} + v_n. \tag{2}$$

Equation (1) calculates a new velocity for each particle (potential solution) based on its previous velocity v_{n-1}, the particle's location at which the best fitness has been achieved x_{ibest}, and the population global (or local neighborhood, in the neighborhood version of the algorithm) location x_{gbest} (or x_{nbest} , in neighborhood version) at which the best fitness so far has been achieved. Equation (2) updates each particle's position in solution hyperspace. The two random control factors ϕ_i $(i = 1, 2)$ are drawn from $U(0, 2.05)$. Moreover, ω is applied to further improve the convergence rate of the PSO, [3].

2.2 Improvement for PSO Algorithm

Improvement for PSO algorithms has been studied extensively in various ways such as increased swarm diversity [4], evolutionary selection [5] and adaptive parameters in the velocity update equations [6]. Although these improvements achieve significant success, they generally obtain superior minima at the expense of iterations. In other words, they concentrate on how to obtain better final fitness but not how to obtain them faster.

Obviously, there is a tradeoff between convergence speed and the values of final fitness for nonlinear optimization methods [4], which means that improving one is at the expense of the other. However, given an order of magnitude for the final solution fitness, it is still possible to obtain satisfied solutions faster [7]. The Kalman swarm (KSwarm) proposed in [7], first tried to address this issue in PSO. In KSwarm, a reformulated PSO update equation is treated as the system dynamic in the state space model, while the observation function is a measurement of the best position each particle has obtained in the past. The results in [7] showed that the KSwarm algorithm has a significant improvement in both convergence speed and final fitness. This research works in [7] shows that the possibility or statistical approaches for improvement in PSO algorithm has a great potential. Inspired by the KSwarm, we propose a novel algorithmic framework to combine the particle filter with PSO algorithm, which can effectively get ride of the heavy extra computation problem incurred by KSwarm[1]. Moreover, rather than learning from the best position it has obtained in the past, the particle in the proposed algorithm learns from the information of the whole probability density function formed by all the particles.

3 Particle Filter

Particle filter, introduced in [8], is a class of importance sampling and resampling techniques designed to simulate from a sequence of probability distributions for sequential inference problems. These methods have gained popularity in recent

[1] The superior performance of the KSwarm is at the cost of additional computation complexity which is incurred by the matrix operations in Kalman update equations whose complexity order is $O(d^3)$ in the number of dimensions [7].

years, due to their simplicity, flexibility, ease of implementation, and modelling success over a wide range of challenging applications [1].

Give a state space model,

$$x_n = f(x_{n-1}) + w_n, \tag{3}$$

$$y_n = h(x_n) + v_n, \tag{4}$$

where $f(\cdot)$ is the system evolution function, $h(\cdot)$ is the observation function, w_n and v_n are the system noise and observation noise respectively. We refer to n as the time index, which is just a simple counter and has not any relationship with 'real' time. In the context of sequential Bayesian inference, we are interested in the posterior distribution $\pi(x_n|y_{1:n})$ (where $y_{1:n}$ denotes the observations $y_1, y_2, ..., y_n$), which can be recursively obtained from the following two equations:

$$\pi(x_n|y_{1:n-1}) \propto \int p(x_n|x_{n-1})\pi(x_{n-1}|y_{1:n-1})dx_{n-1}, \tag{5}$$

and

$$\pi(x_n|y_{1:n}) \propto L(x_n; y_n)\pi(x_n|y_{1:n-1}), \tag{6}$$

where $p(x_n|x_{n-1})$ represents the system dynamic in (3), $\pi(x_{n-1|}y_{1:n-1})$ in (5) is the posterior distribution at $(n-1)$, and $L(x_n; y_n)$ in (6) refers to the likelihood function obtained in (4). The recursion is initialised with some distribution, for example, $p(x_0)$.

In very limited scenarios, the state space models of interest are 'weakly' nonlinear and Gaussian in which one may utilize the Kalman filter and its derivatives [9], to obtain an approximately optimal solution. In practice, it is well known that the update expression in (6) is generally analytically intractable for most models of interest. We therefore turn to sequential Monte Carlo (SMC) methods [1,8], also known as particle filters, to provide an efficient numerical approximation strategy for recursive estimation of complex models.

3.1 Sequential Importance Sampling

The basic idea behind particle filters is very simple: the target distribution is represented by a weighted set of Monte Carlo samples which are called particles in this paper. These particles are propagated and updated using a sequential version of importance sampling as new measurements become available. Hence statistical inferences of the posterior $\pi(x_n|y_{1:n})$ can be computed by these particles.

From a large set of particles $\left\{x_{n-1}^{(i)}\right\}_{i=1}^{N}$ associated importance weights $\left\{w_{n-1}^{(i)}\right\}_{i=1}^{N}$, we approximate the posterior distribution function $\pi\left(x_{n-1}|y_{1:n-1}\right)$ as follows:

$$\pi\left(x_{n-1}|y_{1:n-1}\right) \approx \sum_{i=1}^{N} w_{n-1}^{(i)} \delta\left(x_{n-1} - x_{n-1}^{(i)}\right), \tag{7}$$

where $\delta(\cdot)$ is the Dirac delta function. We would like to generate a set of new particles $\left\{x_n^{(i)}\right\}_{i=1}^{N}$ from an appropriately selected proposal function, i.e.,

$$x_n^{(i)} \sim q\left(x_n|x_{n-1}^{(i)}, y_{1:n}\right), i = \{1, ..., N\}. \tag{8}$$

With the set of state particles $\left\{x_n^{(i)}\right\}$ obtained from (8), the importance weights $w_n^{(i)}$ are recursively updated as follows:

$$w_n^{(i)} \propto w_{n-1}^{(i)} \times \frac{L\left(x_n^{(i)}; y_n\right) p\left(x_n^{(i)}|x_{n-1}^{(i)}\right)}{q\left(x_n^{(i)}|x_{n-1}^{(i)}, y_{1:n}\right)} \tag{9}$$

with $\sum_{i=1}^{N} w_n^{(i)} = 1$. It follows that the new set of particles $\left\{x_n^{(i)}\right\}_{i=1}^{N}$ with the associated importance weights $\left\{w_n^{(i)}\right\}_{i=1}^{N}$ is then approximately distributed according to $\pi\left(x_n|y_{1:n}\right)$.

As the particle filters operate, only a few particles contribute significant importance weights in (9), which leads to a degeneracy problem [10]. To avoid this problem, one possible method is to resample the particles according to the importance weights. With this method, the particles with more significant weights will be selected more frequently than those with less significant weights. More detailed discussions of degeneracy and resampling can be found in [10].

An important element in generating a set of weighted particles which could well approximate the posterior distribution function in (4) is the selection of the proposal importance sampling function $q(x_n^{(i)}|y_{1:n})$ in (8). One choice of the state proposal function is the dynamic prior, $q\left(x_n|x_{n-1}, y_{1:n}\right) = p\left(x_n|x_{n-1}\right)$. Weights become proportional to likelihood, $w_n \propto w_{n-1}L(x_n; y_n)$. In the proposed particle filter optimization algorithm in the following sections, we will restrict to utilize this simple but generic effective proposal.

3.2 Particle Filter for Optimization

Particle filter technique has recently been extended to a general framework to deal with the static and sequential Bayesian inference, as well as the global optimization [11], which is also called sequential Monte Carlo sampler. To apply the SMC sampler for optimization problem, a sequence of artificial intermediate distributions is required, for example $\pi_n(x) = [\pi(x)]^{\tau_n}$ where $\{\tau_n\}_{n=1}^{N}$ is such that $0 < \tau_1 < \cdots < \tau_N$ and $1 << \tau_N$ to ensure that $\pi_0(\cdot)$ is easy to sample from and $\pi_N(\cdot)$ is concentrated around the set of global maxima of $\pi(\cdot)$. Given some sequence of distributions, SMC propagates samples forward from one distribution to the next according to a sequence of Markov kernels, K_n, and corrects for the discrepancy between the proposal and the target distribution by importance sampling [11]. The choice of forward and backward transition kernels is critical in SMC sampler [11]. For example, in [2], the forward transition kernel was chosen as the Markov chain transition kernel from an adaptive Metropolized independence sampler. Due to the efficiency of the adaptive Metropolized independence sampler, the resulting optimization algorithm performs well for combination optimization problems. However, heuristic optimization techniques are not easy to incorporate into the SMC sampler, since it is generic too sophisticated to formulate a heuristic optimization method as the forward and/or backward transition kernel.

In this paper, we provide an alternative way to incorporate heuristic optimization techniques, particularly the population based optimization methods, into the particle filter optimization method. The basic idea is to utilize the desirable tracking ability of particle filter to track the movement of the individuals in the population based optimization algorithms. The location of the global optima is treated as the observation of the dynamic system. Therefore, by treating the location of global optima as the destination, the individuals in the population will move towards it sequentially. Moreover, the move strategy can be very heuristic and efficient since it comes from the excellent population based optimization algorithms.

So the PFO algorithm can be designed as follows: the move strategy in the population based optimization algorithms is reformatted as a system dynamic function. The second step is the definition of observation function. Take the minimization problem $x^* = \arg\min_{x\in\mathcal{X}} g(x)$ as an example. The 'best' observation is of course the exact location of the global minimum of the problem of interest, but it is unknown. If the value of the objective function at the global minimum point is known, i.e. $g(x^*)$, then it can be served as the observation. Therefore we can define the observation function (measurement likelihood) as follows

$$L(x; g(x^*)) = \exp\left\{\frac{-\left[g(x) - g(x^*)\right]^2}{\tau}\right\}, \tag{10}$$

where τ is a properly chosen temperature in the Boltman distribution. However, in plenty of problems, the value of $g(x^*)$ is unknown. In such cases, if we can guess a value g^* which is less than the value of $g(x^*)$, we can define the observation function as,

$$L(x; g^*) = \exp\left\{\frac{-\left[g(x) - g^*\right]}{\tau}\right\}. \tag{11}$$

If we can not even guess such a value, we can also use an observation function as,

$$L(x; a, b) = \exp\left\{\frac{-a * \left[g(x) - b\right]}{\tau}\right\}, \tag{12}$$

where a and b are two constant values, which are properly chosen to make the value of $L(x; a, b)$ reasonable (not extremely large or small).

When the system dynamic and observation in the state space model have been defined, the particle filter is then applied to simulate this model and the particles will move toward the global optima sequentially.

4 Particle Filter Optimization with Particle Swarm Movement

In this section, we will introduce the details of particle filter optimization algorithm. The system dynamic function in the state space model is

$$z_n \triangleq \begin{bmatrix} x_n \\ v_n \end{bmatrix} = \begin{bmatrix} x_{n-1} + wv_{n-1} + \phi_1(x_{ibest} - x_{n-1}) + \phi_2(x_{gbest} - x_{n-1}) \\ wv_{n-1} + \phi_1(x_{ibest} - x_{n-1}) + \phi_2(x_{gbest} - x_{n-1}) \end{bmatrix} + \begin{bmatrix} \epsilon_x \\ \epsilon_v \end{bmatrix} \tag{13}$$

where $\epsilon_x \sim N(0, \Sigma_x)$, $\varepsilon_v \sim N(0, \Sigma_v)$, and $\phi_i \sim U(0, 2.05)$ $(i = 1, 2)$. The observation function can be defined as (10), (11), (12) depending on the concrete problem, here denoted as $L(x; \cdot)$. Description of the particle filter optimization algorithm is presented as follows:

Step 1: At iteration $n = 1$, sample N particles $\left\{z_n^{(i)}\right\}_{i=1}^{N} \sim U(z; \theta)$ $(i = 1, ..., N)$ according to an uniform distribution with a predefined parameter θ (i.e. the parameter to define the feasible solution space of the problem), and compute $w_1^{(i)} \propto L(x_1^{(i)}; \cdot)$.

Step 2: Evolve particles $\left\{z_n^{(i)}\right\}_{i=1}^{N}$ according to the dynamic function (13).

Step 3: Calculate the important weights, $w_n^{(i)} \propto w_{n-1}^{(i)} L\left(x_n^{(i)}; \cdot\right)$.

Step 4: Resample the particle representation $\left\{w_n^{(i)}, z_n^{(i)}\right\}$.

Step 5: Update the location x_{ibest} and x_{gbest}.

Step 6: If the stopping criterion is satisfied, then stop; otherwise, set $n := n + 1$ and go back to step 2.

It is worth noting that in the PFO algorithm, compared with the PSO algorithm, the additional computation is the resampling step, which has the complexity $O(N)$ in the number of particles (N). This additional computation complexity is significantly less than the one in KSwarm whose complexity is about $O(Nd^3)$. Moreover, the number of iterations required by the PFO algorithm is much smaller than those of the PSO and KSwarm algorithm.

5 Experiments

PFO will be compared with the PSO and KSwarm algorithm via the following four benchmark problems: Sphere, DejongF4, Rosenbrock and Griewank. Herein, the first three are unimodal optimization problems, while the last one is multimodal. In all experiments, the dimensionality $d = 30$. The mathematical definitions of four benchmark functions are given as follows:

$$Sphere(x) = \sum_{i=1}^{d} x_i^2, \quad x \in (-50, 50)^d \tag{14}$$

Table 1. Final Values Comparison among PSO, KSwarm and PFO

Function	PSO [7]	KSwarm [7]	PFO
Sphere	370.041	4.723	**5.548e-6**
DejongF4	4346.714	4.609	**2.236e-9**
Rosenbrock	2.61e7	3.28e3	**3.16e-2**
Griewank	13.865	0.996	**2.845e-5**

$$DeJongF4(x) = \sum_{i=1}^{d} i x_i^4, \quad x \in (-20, 20)^d \tag{15}$$

$$Rosenbrock(x) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right], \quad x \in (-100, 100)^d \tag{16}$$

$$Griewank(x) = \frac{1}{4000} \sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1, \quad x \in (-600, 600)^d \tag{17}$$

In all experiments, the particle size is set to 100. The stopping criterion is that the algorithm reaches its maximum iteration number 200. Then, the number of objective function evaluations is the same as it is in [7], which makes the comparison reasonable. We run each experiment 50 times for 200 iterations, and averaged the results to account for stochastic differences. The inertia weight w is 0.5 in all the experiments. The variance of system noise is given as $\varSigma_x = \gamma I_d$, $\varSigma_v = \gamma I_d$, where the scalar γ indicates the magnitude of the variance in each dimension, specifically, $\gamma = 0.001$ when iteration number $n \leq 100$ and $\gamma = 0.00001$ when $n > 100$ for all experiments. A large variance enables the algorithm to explore the space quickly while a small variance enables the algorithm to improve the final fitness. In design of the observation function, we assumed that we have only 'weak' prior knowledge of the value of the optimal objective function $g(x^*)$. For instance, in all the experiment, we used the observation function (12) by simply setting $a = 1$ and $b = 0$. The temperature in the Boltzmann distribution τ is set by experience. Herein, in all the experiments, $\tau = 10$.

Table 1 shows the final values of basic PSO and KSwarm algorithms after 1000 iterations, as well as the final values of PFO after 200 iteration. It can be seen that the values obtained by PFO are several orders of magnitude better than the basic PSO and KSwarm algorithms in all four benchmark problems. Afterwards, Figs. 1 - 4 pictorially depict the best value versus the number of iterations. Herein, in order to compare the PFO, PSO and KSarm under the same complexity condition, the number of iterations for PFO is set to 200, while the number of iterations for PSO and KSwarm is set to 1000. Although the number of iterations is different, the number of time of calculating the objective function is same. Through simulations, we observe that, compared with the PSO algorithm, the additional computation efforts of PFO is negligible, while KSwarm take significant additional computation due to its matrix operations.

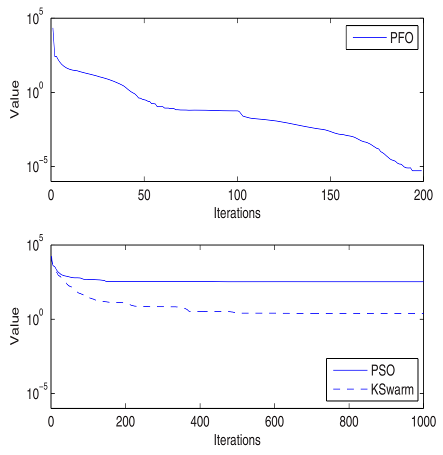


Fig. 1. Sphere

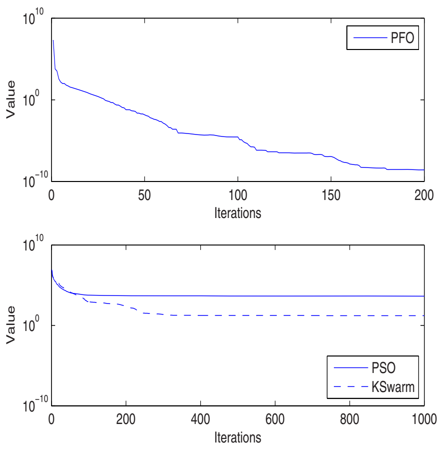


Fig. 2. Dejong

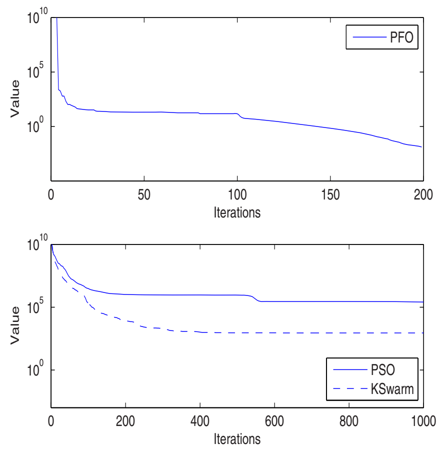


Fig. 3. Rosenbrock

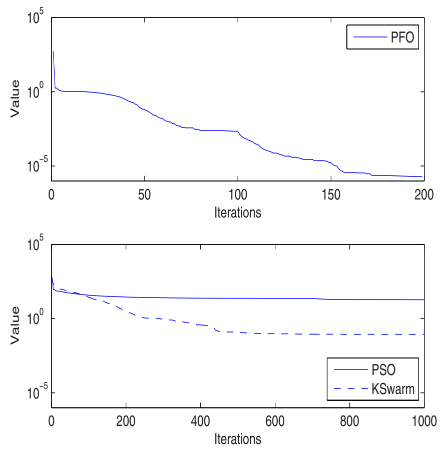


Fig. 4. Griewank

Moreover, the plots of the best value obtained per iteration show that the PFO tends to find good solutions faster than the other two methods. These simulation results demonstrate a significant improvement for the PSO, not only in exploring final solutions, but also in the speed to find them.

6 Conclusions

In this paper, we propose a novel generalized framework for stochastic optimization in which the PFO is incorporated with the swarm move method in PSO. The particle swarm move in PSO algorithm is treated as the system dynamic in the state space model, while the objective function in optimization problem is designed as the observation model. Particle filter method is then applied to track the dynamic movement of the particle swarm in PSO algorithm. By incorporating the state space probability modelling and resample strategy into the PSO algorithm, the PFO can potentially enhance the ability of PSO algorithm in two aspects: making it easier to jump out local optima and refining the final result. Compared with the PSO and KSwarm algorithm, the PFO algorithm can obtain better final fitness with a negligible additional computation effort. Finally, simulation results demonstrate a significant improvement for the PFO, not only in exploring final solutions, but also in the speed to find them.

Acknowledgment

The work by Shengxiang Yang was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/1.

References

1. Doucet, A., De Freitas, J., Gordon, N. (eds.): Sequential Monte Carlo Methods in Practice. Springer, Heidelberg (2001)
2. Zhang, Y., Ji, C., Walik, W.Q., Liu, Y., O'Brien, D.C., Edwards, D.J.: Joint Antenna and User Selection Algorithm for Uplink of Multiuser MIMO Systems Using the Sequential Monte Carlo Optimization. In: IEEE Workshop on Statistical Signal Processing, SSP 2007, Madison, Wisconsin, pp. 493–496. IEEE Press, Los Alamitos (2007)
3. Shi, Y., Eberhart, R.: Empirical Study of Particle Swarm Optimization. In: IEEE Congress on Evolutionary Computation, CEC 1999, pp. 1945–1950. IEEE Press, Piscataway (1999)
4. Riget, J., Vesterstroem, J.S.: A diversity-guided Particle Swarm Optimizer - the ARPSO. Department of Computer Science, University of Aarhus, Technique Report (2002)
5. Angeline, P.J.: Using Selection to Improve Particle Swarm Optimization. In: IEEE Congress on Evolutionary Computation, CEC 1998, Anchorage, Alaska, USA, pp. 84–89. IEEE Press, Los Alamitos (1998)
6. Clerc, M., Kennedy, J.: The particle Swarm-explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
7. Monson, C.K., Seppi, K.D.: The Kalman Swarm. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 140–150. Springer, Heidelberg (2004)
8. Gordon, N., Salmond, D., Smith, A.: Novel Approach to Nonlinear/ Non-gaussian Bayesian State Estimation. IEE Proceedings Radar and Signal Processing 140(2), 107–113 (1993)
9. Harvey, A.C.: Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press, Cambridge (1990)
10. Doucet, A., Godsill, S., Andrieu, C.: On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. Statistics and Computing 10, 197–208 (2000)
11. Moral, P.D., Doucet, A., Jasra, A.: Sequential Monte Carlo Samplers. Royal Statistical Society 68, 411–436 (2006)

A Feasibility-Preserving Crossover and Mutation Operator for Constrained Combinatorial Problems

Martin Lukasiewycz, Michael Glaß, and Jürgen Teich

Hardware-Software-Co-Design
Department of Computer Science 12
University of Erlangen-Nuremberg, Germany
{martin.lukasiewycz,glass,teich}@cs.fau.de

Abstract. This paper presents a feasibility-preserving crossover and mutation operator for evolutionary algorithms for constrained combinatorial problems. This novel operator is driven by an adapted Pseudo-Boolean solver that guarantees feasible offspring solutions. Hence, this allows the evolutionary algorithm to focus on the optimization of the objectives instead of searching for feasible solutions. Based on a proposed scalable testsuite, six specific testcases are introduced that allow a sound comparison of the feasibility-preserving operator to known methods. The experimental results show that the introduced approach is superior to common methods and competitive to a recent state-of-the-art decoding technique.

1 Introduction and Related Work

Definition 1. *A* constrained combinatorial problem *is defined as:*

$$\begin{aligned}&\textit{minimize } f(x)\\&\textit{subject to } x \in X_f \textit{ with } X_f \subseteq \{0,1\}^n\end{aligned}$$

The objective function f allows multi-dimensional and non-linear calculations. The search space $X = \{0,1\}^n$ is restricted to binary values, but allows integer values by a binary encoding. The *feasible search space* $X_f \subseteq X$ is restricted by a set of linear constraints which are subsumed in the following matrix inequation:

$$Ax \leq b \tag{1}$$

with $A \in \mathbb{Z}^{m,n}$ and $b \in \mathbb{Z}^m$. Thus, the constraints have to be linear or linearizable[1]. Constraints that are not linearizable have to be handled by common methods like penalty functions. However, many problems have linear constraints only [1] or are dominated by the number of linear constraints [2]. In the following, this paper assumes that all constraints are linear.

In the field of *Evolutionary Computation* different constraint-handling techniques of constrained combinatorial problems as stated in Definition 1 exist. These constraint-handling methods become necessary since a variation by crossover and mutation

[1] Linearization substitution rule: $x_1 \cdot x_2 \leftrightarrow x_3$ with $x_1 - x_3 \geq 0 \wedge x_2 - x_3 \geq 0 \wedge x_1 + x_2 - x_3 < 2$.

operators tends to deliver infeasible solutions. A straightforward approach is a *penalty function* that counts the violated constraints and deteriorates the objective function, cf. [3,4]. Moreover, a greedy repair algorithm as presented in [5] can be applied to minimize the number of violated constraints. This repair algorithm delivers feasible solutions only for specific problems like, e.g., the *0/1 Knapsack Problem* [5], but it cannot guarantee feasibility for linear constraints as stated in Equation (1). Since this problem is known to be NP-complete in general [1], it cannot be solved by a greedy algorithm (assuming P$\neq$NP).

If the search space is hard constrained, the common constraint-handling methods fail, since they are more focused on the search for feasible solutions than optimizing the objectives. In [2], a decoding strategy [6] known as *SAT decoding* is presented that overcomes these drawbacks. By using a *Pseudo-Boolean (PB) solver* [7] as a decoder, this method always obtains feasible solutions by mapping from a bounded search space to feasible solutions. In combination with an *Evolutionary Algorithm* (EA), a good convergence towards the optimal solutions also on large and complex real-world problems [8] is reached.

Instead of using the PB solver for decoding feasible solutions, our novel approach uses the PB solver within the crossover and mutation operator. Thus, only feasible solutions are obtained already in the variation process of the EA and a decoding becomes unnecessary. To compare this novel approach to common constraint-handling techniques and the SAT decoding, we present a testsuite and six specific testcases. These testcases represent random as well as structured real-world problems.

The remainder of the paper is outlined as follows: Section 2 introduces the scalable testsuite and six specific testcases for the constrained combinatorial problem. In Section 3, the novel feasibility-preserving crossover and mutation operator is presented. Experimental results are given and discussed in Section 4 before the paper is concluded in Section 5.

2 Testsuite

In the following, a scalable testsuite for constrained combinatorial problems is presented. This testsuite is based on the well known *3-SAT* problem [1]. The 3-SAT problem is a special case of the *Satisfiability Problem* where each clause contains exactly three literals. In a nutshell, the 3-SAT problem is to determine if there exists a satisfying solution to a Boolean formula given as a conjunction of clauses. A clause is a disjunction of literals, which are variables or their negations. Each clause can be formulated as a linear greater-or-equal-1 constraint by substituting the *ORs* by plus signs and the negative literals $\overline{x_i}$ by $1 - x_i$. For instance, the clause $(x_1 \vee x_2 \vee \overline{x_3})$ is converted to $x_1 + x_2 + (1 - x_3) \geq 1$ or $x_1 + x_2 - x_3 \geq 0$, respectively. Thus, a transformation to the form in Equation (1) is straightforward.

The testsuite is configured by two parameters:

- n - number of binary variables
- k - number of constraints

In contrast to the original 3-SAT problem, the constructed constrained problem must contain at least one feasible solution. This is ensured by first constructing a random solution $x \in X$ with $\sum_{i=1}^{n} x_i = \frac{n}{2}$ or the same number of $1s$ and $0s$, respectively. Random constraints are generated by constructing clauses with randomly chosen positive (x_i) or negative literals ($\overline{x_i}$) of three randomly selected variables. This generated constraint is only added if it is satisfied by x to guarantee at least a single feasible solution. The procedure is carried out until the desired number of constraints k is reached.

Since this paper deals with constraint-handling, for all following testcases, the objective function f is a simple single linear objective function

$$f(x) = \sum_{i=1}^{n} r_i \cdot x_i$$

with the r_i values randomly distributed in $\mathbb{R}_{[0,1]}$.

T1. Testcase T1 is a random instance with $n = 100$ and $k = 400$. With $\frac{k}{n} \approx 4.3$, hard solvable 3-SAT instances are generated, cf. [9]. The experience shows, $\frac{k}{n} = 4$ tends to construct testcases that have enough feasible solutions for a meaningful optimization but, on the other hand, obtaining a feasible solution is not trivial.

T2. Testcase T2 is a random instance with $n = 150$ and $k = 600$. This testcase, in combination with T1, reveals the scaling of the optimization method for random instances.

T3. Testcase T3 represents a real-world problem with $n = 1000$ and $k = 4000$. For this testcase, a structure is induced by a special scheme to select the variables for the clauses: First, a random variable x_i is selected. The other two variables x_j,x_k are selected by a *Geometric Distribution* with the inverse function $F^{-1}(u) = \lceil ln(1-u)/ln(1-p) \rceil$, with u being a random number in $\mathbb{R}_{[0,1)}$, such that $j = (i + F^{-1}(u))\%n + 1$ and $k = (j + F^{-1}(u))\%n + 1$. A high value for p decreases the pairwise *distance* of the variables that is in average $\frac{1}{p}$. For testcase T3 the value p is set to 0.1. It is known that PB solvers are performing much better on structured problems than on random generated problems [10]. In fact, real-world problems are usually structured due to the problem trait, like, e.g. the system structure, or the construction scheme like, e.g, a hierarchical approach, cf. [9].

T4 Testcase T4 represents a real-world problem with $n = 1000$ and $k = 4100$. Unlike the construction scheme used in T3, a structure is induced by a partitioning approach. First, b constraints are randomly generated among all variables. Second, the variables are uniformly distributed in a partitions such that for each partition $\frac{k-b}{a}$ random constraints are generated. For testcase T4, the parameters are set to $a = 10$ and $b = 100$ and, thus, 10 instances as in testcase T1 are created and interconnected by 100 constraints. These additional 100 constraints further reduce the rate of feasible solutions.

T5. Testcase T5 is constructed the same way as T4 with $n = 2000$, $k = 8100$, $a = 20$ partitions, and $b = 100$. This testcase is used to illustrate the scaling of the optimization methods on structured problems.

T6. Testcase T6 is constructed the same way as T3 with $n = 3000$ and $k = 9000$. With $\frac{k}{n} = 3$ this testcase has a high rate of feasible solutions. This testcase shows the range of applicability of the optimization methods for low constrained problems.

3 Feasibility-Preserving Crossover and Mutation

The requirement for a feasibility-preserving crossover and mutation operator for constrained combinatorial problems is to always obtain a feasible offspring solution from two feasible parent solutions. The proposed approach is based on a state-of-the-art *Pseudo-Boolean (PB) solver* [7]. In the following, a specialized crossover and mutation operator is presented that makes use of a PB solver to obtain feasible offspring solutions from two feasible parent solutions. Moreover, a heuristic that aims to improve the quality of the obtained offspring solution in terms of information preservation is presented.

3.1 PB Solver

The task of a PB solver is to find an $x \in X_f$ that satisfies a set of linear constraints as formulated in Equation (1). In fact, this NP-complete problem [1] is an ILP with binary variables and an empty objective function and can be solved by a common ILP solver. However, the specialized PB solvers tend to outrun common ILP solvers on these Boolean-natured problems [11]. These PB solvers are extended *SAT* solvers that are actually used to solve the *Satisfiability problem* and are based on a backtracking strategy. This strategy is known as the DPLL algorithm [12] and is outlined in Algorithm 1. The algorithm efficiently searches for a solution $x \in X_f$ that fulfills all given constraints, cf. [7]:

Algorithm 1. DPLL backtracking algorithm: **solve**

```
Require: $\rho \in \mathbb{R}^n, \sigma \in \{0,1\}^n$
Ensure: $x \in X_f$
1: while true do
2:     branch($\rho, \sigma$)
3:     if CONFLICT then
4:         backtrack()
5:     else if SATISFIED then
6:         return $x$
7:     end if
8: end while
```

Starting with completely unassigned variables, the operation $branch(\rho, \sigma)$ chooses an unassigned variable and assigns it a value (line 2). The rule which variable is chosen and which value is assigned is called *branching strategy*. The branching strategy is guided by the two vectors $\rho \in \mathbb{R}^n$ and $\sigma \in \{0,1\}^n$. Unassigned variables x_i with the highest value ρ_i are prioritized and are set to the value σ_i. After each variable assignment, conflicts are recognized (line 3). If any constraint is not satisfiable anymore, the backtracking is triggered (line 4), i.e., variable assignments are reverted. In case all variables have an assignment and there exists no conflict (line 5), this assignment represents a feasible solution x which is returned (line 6).

3.2 Feasibility-Preserving Operator

Algorithm 1 is able to find feasible solutions $x \in X_f$. The backtracking is guided by the branching strategy (ρ, σ) and, thus, these two vectors have a high influence on which solution in X_f is found. Thus, Algorithm 1 is used to generate a feasible initial population by arbitrary random branching strategies.

In Algorithm 2, a feasibility-preserving crossover and mutation operator is presented: Based on the two feasible parent solutions, a branching strategy is derived to obtain a feasible offspring solution using Algorithm 1.

Algorithm 2. Feasibility-preserving crossover and mutation operator

```
Require: $x', x'' \in X_f$; $C \subseteq \{1, ..., n\}$; $r \in \mathbb{R}_{[0,1]}$ (mutation rate)
Ensure: $x \in X_f$
 1: for $i \in \{1, ..., n\}$ do
 2:     if $i \in C$ then
 3:         $\sigma_i = x'_i$
 4:     else
 5:         $\sigma_i = x''_i$
 6:     end if
 7:     $\rho_i = rand(0, 1)$
 8:     if $rand(0, 1) < r$ then
 9:         $\rho_i = \rho_i + 1$
10:         $\sigma_i = \overline{\sigma_i}$
11:     end if
12: end for
13: $x = \mathbf{solve}(\rho, \sigma)$
14: return $x$
```

Algorithm 2 requires two feasible parent solutions $x', x'' \in X_f$, the selection set $C \subseteq \{1, ..., n\}$, and the mutation rate r. A branching strategy (ρ, σ) is generated as follows: The prioritized phase σ_i of the corresponding variable x_i is set to the corresponding value of one of the parent solutions x'_i or x''_i, respectively. This selection is done based on the set C that controls the binary crossover (line 2-6). The priority ρ_i of a variable x_i is randomly chosen in $\mathbb{R}_{[0,1]}$ (line 7). For each variable x_i a mutation is done with the probability r. The mutation increases the priority ρ_i by 1 and flips the prioritized phase value σ_i (line 8-11).

With the given branching strategy (ρ, σ) the PB solver finds a feasible offspring solution $x \in X_f$ (line 13). At this, preserving the information obtained by the parent solutions becomes important. Hence, the *adaption diversity* $div(x, \sigma) = \sum_{i=1}^{n} \frac{|x_i - \sigma_i|}{n}$, that measures the fraction of preserved information of σ, should be kept as small as possible.

Theorem 1. *Given two feasible solutions $x, \tilde{x} \in X_f$, a specific σ, and a random ρ, the probability P to obtain the solutions x in comparison to $\tilde{x}$ is*

$$P(x = \mathbf{solve}(\rho, \sigma)) > P(\tilde{x} = \mathbf{solve}(\rho, \sigma)) \quad (2)$$

if

$$div(x, \sigma) < div(\tilde{x}, \sigma). \tag{3}$$

Proof. Equation (3) implies that $\tilde{x}$ compared to x has a higher count of variables that are different to these corresponding variables of σ. Thus, with a given random ρ, $\tilde{x}$ is excluded with a higher probability earlier in the backtracking search algorithm $\mathbf{solve}(\rho, \sigma)$ compared to x. Hence, x is reached with a higher probability than $\tilde{x}$ as stated in Equation (2).

Thus, the PB solver tends to find a similar offspring solution x compared to σ and preserves the information passed along by the parent solutions x' and x''. By setting C to $\{1, ..., n\}$ or $\{\}$, respectively, Algorithm 2 decays to a feasibility-preserving mutation only operator. This approach is applicable if the *crossover rate* is lower than 1.

3.3 Minimizing the Adaption Diversity

To preserve the information within the feasibility-preserving operator, the adaption diversity has to be minimized. At this, the selection set C that controls the binary crossover has a high influence on this value. We propose a heuristic that finds selection sets based on a graph that represents the constraints of the problem. On the one hand, this heuristic decreases the adaption diversity and, on the other hand, decays to the well known one-point crossover for an unconstrained problem.

Consider the following definitions:

Definition 2 (Constraint-Graph). *A* constraint-graph $G(V, E)$ *is an undirected graph that contains a vertex i for each variable x_i from a problem defined in Equation (1). A function $w : V \times V \to \mathbb{R}$ defines the weight of the edges. For each constraint of the problem an edge between each pair x_i, x_j of variables of the constraint*[2] *is added between the vertices i and j. The weight of the added edge $w(i, j)$ is the reciprocal value of the count of the variables of the constraint. In case there exists already an edge between the vertices i and j, the calculated weight is added to the weight of the existing edge.*

Definition 3 (Cut). *Let $G(V, E)$ denote a graph. A* cut *is a partition of the vertices V in two disjunctive sets C and $\overline{C}$. Any edge $e = (u, v) \in E$ with $u \in C$ and $v \in \overline{C}$ is a* cut edge. *A* weight *of a cut is the sum of the weights of the cut edges.*

A cut on a constraint-graph produces two partitions C and $\overline{C}$. At this, C can be used as the selection set for Algorithm 2. A small cut weight should be aspired since it tends to minimize the number of potentially conflicting constraints and, thus, also tends to minimize the adaption diversity.

A *min-cut algorithm* that finds the minimal cut of an undirected graph is presented in [13]. However, a reasonable crossover operator needs a sufficient number of cuts instead of just a single minimal cut. Based on Algorithm 3, the following proposed heuristic tends to generate $n - 1$ relatively small cuts.

[2] Variables are said to be part of a constraint if their corresponding coefficient in C from Equation (1) is non-zero.

Algorithm 3. Vertex ordering heuristic

```
Require: $G(V, E), w$
Ensure: $P$ is an ordered set
1: while $P \neq V$ do
2:     Select $x_i \in V \backslash P$ with $\sum_{x_j \in P} w(x_i, x_j) = max\{\sum_{x_k \in P} w(x_i, x_k) | x_k \notin P\}$
3:     $P = P \cup \{x_i\}$
4: end while
5: return $P$
```

Given the constraint-graph $G(V, E)$ with the corresponding edge weight function w, the algorithm fills a set P with the vertices V and keeps track of the insertion order. Until the set P contains all vertices from V the algorithm continues (line 1). Each step the most tightly connected vertex x_i with respect to the set P is added to P (line 2-3).

Splitting the ordered set P at one point into two subsets C and $\overline{C}$ generates $n - 1$ cuts with relatively small weights. This is due to the fact that by using the presented heuristic and adding at each step the most tightly connected edge, the weights of the cuts are kept small along the order of P. Thus, we will uses these $n - 1$ C subsets for the selection set C for Algorithm 2. In fact, this approach is similar to the common one-point crossover and decays to it for unconstrained problems.

The time complexity of this algorithm is $O(|E| + |V|log|V|)$, as stated in [13], and with $|V| = n$ and a maximum value of $|E| = n^2$, the aggregated worst-case complexity is $O(n^2)$. However, this algorithm has to be performed only once for each problem. As the experimental results validate, the costs of this heuristic are negligible small compared to the overall runtime of one optimization.

4 Experimental Results

All experimental results were carried out on an Intel Pentium 4 3.20 GHz machine with 1 GB RAM. The implementation of the presented approach is based on the optimization framework OPT4J [14].

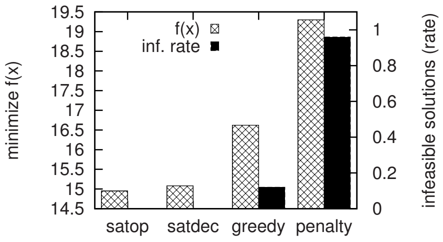
4.1 Selection Set

Table 1 presents a comparison of the adaption diversity induced by the feasibility-preserving operator for a completely random selection set C and the selection sets that were obtained by the presented heuristic in Section 3.3.

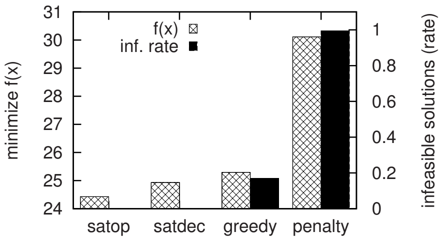
In particular, the induced adaption diversity for the structured testcases T3-T6 is significantly lower for the sets that are obtained by the proposed heuristic. These results

Table 1. Results for the adaption diversity on all testcases with a *random* selection set C and random selection set that was obtained by the presented *heuristic*

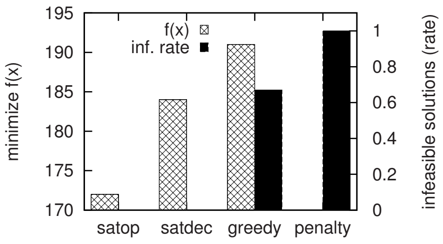
	T1	T2	T3	T4	T5	T6
random	0.19	0.22	0.19	0.10	0.16	0.18
heuristic	0.12	0.14	0.04	0.02	0.02	0.01



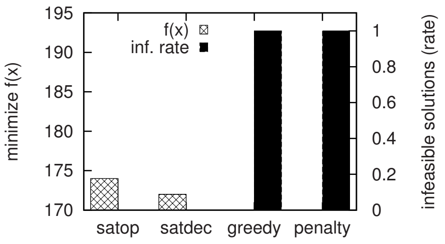
(a) Results for T1.



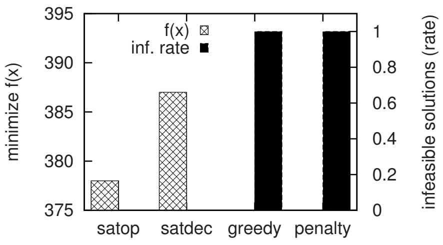
(b) Results for T2.



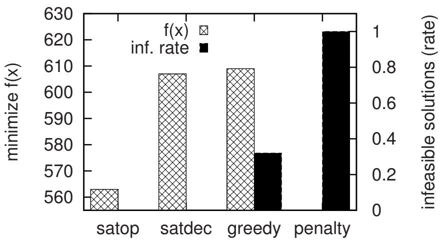
(c) Results for T3.



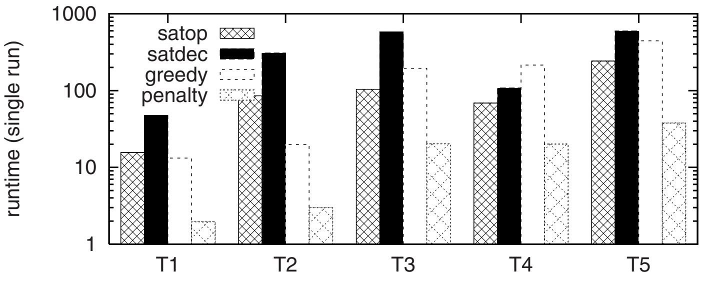
(d) Results for T4.



(e) Results for T5.



(f) Results for T6.



(g) Runtimes for all testcases.

Fig. 1. 1(a) to 1(f) show the results of the four optimization methods on the presented testcases. Given is the reached minimal value of $f(x)$ as well as the rate of infeasible solutions that were obtained throughout the optimization. In 1(g), the runtimes of the optimization methods on the testcases are denoted. Note that these values are given in logarithmic scale.

validate that the heuristic effectively preserves the information of the parents. Thus, for following experimental results the novel feasibility-preserving technique is performed in combination with the presented adaption diversity minimizing heuristic.

4.2 Optimization

The section compares different *EA* based constraint-handling strategies on the six testcases. The penalty function based approach (*penalty*) tries to minimize the following function (cf. Equation (1)) that prioritizes feasible over infeasible solutions:

$$f'(x) = f(x) + p(x)$$
$$\text{with } p(x) = n \cdot min\{e_1 + ... + e_n\} \text{ such that } Ax \leq b + e \text{ and } e \in {\mathbb{N}_0}^m$$

This approach is extended a by a greedy repair algorithm (*greedy*) [5] that flips each variable trying to minimize $p(x)$. In case of $p(x) = 0$, the greedy algorithm stops since x is a feasible solution. The SAT decoding approach [2] is in the following denoted as *satdec*. The feasibility-preserving approach proposed in this work is denoted as *satop*. For all methods, an elitist EA is used with the population size of 100 individuals, generating 25 offspring from 25 random selected parent solutions. For all methods, the mutation rate is set to $r = \frac{1}{n}$. For the binary vectors, a binary crossover is used, followed by a bit-flip with probability r. For the *satdec* approach, the crossover for the real vector is implemented by the SBX ($\nu = 15$) operator, followed by polynomial mutation ($\eta = 20$) with probability p. For each testcase 10 instances were generated, and for each instance 10 runs were carried out to allow a calculation of am overall meaningful average.

The results of the optimization runs is given in Figure 1. The proposed feasibility-preserving approach *satop* delivers the best solutions for all testcases except T4 where *satdec* is slightly better. This is due to the fact, that *satdec* works best one problems with very few feasible solutions. This is also apparent on testcase T6 with many feasible solutions, where *satdec* is only slightly better than the *greedy* method. However, though *satop* and *satdec* are both based on a PB solver and deliver feasible solutions only, *satop* is about four times faster in average compared to *satdec* and even faster than the *greedy* approach on the structured testcases T3-T6. Except for T4, the *satop* method delivers remarkably better solutions on the testcases T3,T5, and T6 that represent real-world problems. Note that the *greedy* approach becomes remarkably slow on the large problem where obtaining a feasible solution is difficult and fails to find a single feasible solution on the testcases T4 and T5. The method *penalty* is the fastest, but delivers bad results and fails completely to find feasible solutions on the larger problems T3 to T6.

5 Conclusion

In this paper, a feasibility-preserving crossover and mutation operator for constrained combinatorial problems is presented. This operator allows an Evolutionary Algorithm to perform efficiently also on large and problems with few feasible solutions. The experimental results compare this novel approach to known methods based on a proposed testsuite. The results show that the feasibility-preserving operator is superior to common methods like penalty functions or a greedy repair algorithm. Compared to the state-of-the-art SAT decoding, the novel approach is four times faster and delivers, except for one testcase, better solutions.

References

1. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)
2. Lukasiewycz, M., Glaß, M., Haubelt, C., Teich, J.: SAT-Decoding in Evolutionary Algorithms for Discrete Constrained Optimization Problems. In: Proceedings of CEC 2007, pp. 935–942 (2007)
3. Coello, C.: Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. Art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
4. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4(1), 1–32 (1996)
5. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
6. Koziel, S., Michalewicz, Z.: A decoder-based evolutionary algorithm for constrained parameter optimization problems. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 231–240. Springer, Heidelberg (1998)
7. Chai, D., Kuehlmann, A.: A fast pseudo-boolean constraint solver. In: Proceedings of DAC 2003, pp. 830–835 (2003)
8. Lukasiewycz, M., Glaß, M., Haubelt, C., Teich, J.: Efficient symbolic multi-objective design space exploration. In: Proceedings of the ASP-DAC 2008, pp. 691–696 (2008)
9. Aloul, F.A., Ramani, A., Markov, I.L., Sakallah, K.A.: Solving difficult SAT instances in the presence of symmetry. In: Proceedings of DAC 2002, pp. 731–736 (2002)
10. Prasad, M.R., Chong, P., Keutzer, K.: Why is ATPG easy? In: Proceedings of DAC 1999, pp. 22–28 (1999)
11. Aloul, F.A., Ramani, A., Markov, I.L., Sakallah, K.A.: Generic ILP versus specialized 0-1 ILP: an update. In: Proceedings of ICCAD 2002, pp. 450–457 (2002)
12. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM 5(7), 394–397 (1962)
13. Stoer, M., Wagner, F.: A simple min-cut algorithm. J. ACM 44(4), 585–591 (1997)
14. Opt4J (Java Optimization Framework), `http://www.opt4j.org/`

An Iterated Local Search Approach for Finding Provably Good Solutions for Very Large TSP Instances

Peter Merz[1,⋆] and Jutta Huhse[2]

[1] Department of Computer Science
University of Kaiserslautern, Germany
peter.merz@ieee.org
[2] IBFI Schloss Dagstuhl
Wadern, Germany
jutta.huhse@dagstuhl.de

Abstract. Meta-heuristics usually lack any kind of performance guarantee and therefore one cannot be certain whether the resulting solutions are (near) optimum solutions or not without relying on additional algorithms for providing lower bounds (in case of minimization).

In this paper, we present a highly effective hybrid evolutionary local search algorithm based on the iterated Lin-Kernighan heuristic combined with a lower bound heuristic utilizing 1-trees. Since both upper and lower bounds are improved over time, the gap between the two bounds is minimized by means of effective heuristics. In experiments, we show that the proposed approach is capable of finding short tours with a gap of 0.8% or less for TSP instances up to 10 million cities. Hence, to the best of our knowledge, we present the first evolutionary algorithm and meta-heuristic in general that delivers provably good solutions and is highly scalable with the problem size. We show that our approach outperforms all existing heuristics for very large TSP instances.

1 Introduction

The Traveling Salesman Problem (TSP) is one of the best-known combinatorial optimization problems. Simply stated, the problem is to find the shortest round trip through a set of cities where each city has to be visited exactly once. Unfortunately, the TSP is known to be NP-hard.Meta-heuristics usually do not provide a performance guarantee such as approximation algorithms. Hence, for finding provably good solutions, one has to resort to exact algorithms like Branch & Cut, or use an algorithm for computing a lower bound (in case of a minimization problem) additionally to the meta-heuristic.

In this paper, we present a heuristic approach that simultaneously improves lower and upper bounds for a TSP instance to provide a gap for the best solution found. The gap determines the maximum deviation from the optimum solution

⋆ This work is partially supported by the German Research Foundation (DFG).

and therefore provides a quality measure for the obtained TSP tour. The approach differs from exact algorithms like Branch & Cut [1] in that no efficient linear programming (LP) solver is required and it differs from approximation algorithms such as PTAS [2] in that no general performance guarantee is provided. Instead, the quality is proved for each instance and a particular run: a final gap between lower and upper bound of 1% means that the solution found is at most one percent above the optimum (in practice the real gap is much lower).

We show in experiments that our approach (a) delivers solutions known to be only about 1% above the optimum on average, (b) scales linearly for random Euclidean instances and is therefore even applicable to instances with 10 million cities, and (c) outperforms all known TSP heuristics for very large instances.

The paper is organized as follows: Section 2 discusses state-of-the-art meta-heuristics for the TSP. In Section 3, our new highly effective approach is presented. Results from several experiments are discussed in Section 4. The paper is concluded in Section 5.

2 Effective Approaches for the TSP

The TSP has served as a test-bed for new heuristic approaches including evolutionary algorithms (EA). Consequently, many approaches, both evolutionary and non-evolutionary, have been proposed. Here, we focus on those approaches which are highly effective and scalable. TSP instances up to a size of 1,000 can be considered as trivial for most algorithms. In fact, these small problems can usually be solved exactly by Branch & Cut [3] in a few seconds. Therefore, these instances are no longer of interest for heuristics research on the TSP. For instances up to approx. 30,000 cities, very effective heuristics have been proposed most of which are based on the powerful Lin-Kernighan (LK) heuristic [4], a variable k-opt local search. An example is Helsgaun's LK implementation (LKH) [5].

Only few evolutionary algorithms can compete with LKH. One of the best evolutionary approaches is the EA of Nagata using EAX crossover [6] and 2-opt local search. This algorithm finds (near) optimal tours up to a size of 33,000 cities, although with a high runtime. Recently, Nguyen *et al.*[7] have proposed a hybrid evolutionary algorithm which utilizes a variant of the MPX crossover operator [8] and a Lin-Kernighan local search variant with 5-opt moves. Results are reported for instances up to 85,900 cities. The authors claim that their algorithm is more effective than LKH. Moreover, the authors describe an approach for solving the World TSP (approx. 2 million cities) by solving and merging subproblems. But results for other instances in the range from 100,000 to 10 million cities are not reported.

For instances larger than 100,000 cities, only few heuristics have been proposed. For these instances, the DIMACS TSP implementation challenge [9] lists several approaches of which the best are based on the LK heuristic: The multi-level algorithm of Walshaw [10] first reduces the size of a TSP instance stepwise and then applies the (chained) LK heuristic to the smaller problems. The results are inferior to the results obtained by directly applied chained LK or iterated LK heuristics.

These heuristics are based on the principle of iterated local search [11], an evolutionary heuristic incorporating local search. The idea is to stepwise improve the current best solution by mutating it and subsequently applying local search. The first iterated local search was the iterated Lin-Kernighan (ILK) heuristic by Johnson [12]. Other variants have been proposed such as the chained Lin-Kernighan heuristic [13,14]. These ILK heuristics have been applied to instances with up to 10 million cities. The only algorithm within the DIMACS challenge not using LK as a subroutine and still being highly effective for large instances is the dynamic programming approach of Balas and Simonetti [15].

Except for the LKH heuristic, none of the mentioned algorithms provides a lower bound on the optimum solution. To the best of our knowledge, the only evolutionary algorithm computing lower bounds is the one proposed in [16]. However, the approach deals with instances below 2,400 cities only.

3 A Scalable Evolutionary Algorithm for the TSP

In the following, we present an ILK variant for the TSP that can be applied to instances with millions of cities.

3.1 The General Evolutionary Framework

The evolutionary framework we use in our algorithm is not specific to the TSP. The concept of iterated local search has been applied to other combinatorial problems with great success [11]. The framework is rather simple: First, a solution to the problem is generated using some sort of randomized construction heuristic. Then, a local search is applied to obtain a local optimum. Afterwards, the best local optimum obtained so far is mutated repeatedly by some problem–specific mutation operator, and a local search procedure is applied subsequently. If the newly obtained solution is better than the previous one, it is accepted as the new best solution. In this way, one can obtain successively better solutions. The reason why this approach is so effective for the TSP is that the fitness landscape of the TSP is highly correlated: The smaller the distance to the optimum, the better the fitness (the smaller the tour length in case of the TSP). Therefore, iterated local search allows to 'jump' from one local optimum to a better local optimum until the global optimum is reached. Relatively small mutations are necessary to jump to a new local optimum since local optima are close to each other [17,18].

Our local search is based on the LK heuristic, hence our iterated local search is called iterated LK. The general outline of our iterated LK is shown in Fig. 1. In contrast to other approaches, our ILK incorporates a lower bound computation. This computation is interleaved with the optimization algorithm as can be seen in the figure: every 400 iterations of the ILK, the lower bound is improved until there appears to be no more improvement possible (the lower bound computation has converged). The lower bound computation possibly modifies the candidate edge set, which is used by the local search to look for improving moves (edge exchanges).

3.2 Implementation Details of the ILK

To find initial solutions (Init() in the pseudo code), we use the Quick-Boruvka heuristic [9,14], and the initial candidate set (FindInitialCandidateSet(Instance) in the pseudo code) is based on a subgraph containing the two nearest neighbors for each quadrant of a city [14]. This candidate set has the property of being connected. The candidate set is computed using a k-d-tree data structure [19]. A small candidate set is essential for the scalability of the approach. Having eight neighbors on average appeared to be reasonable.

Due to the complexity of state-of-the-art implementations of ILK, it is not possible to describe all the aspects here in fully detail. A forthcoming technical report will cover all these aspects.

Mutation Operator. The mutation operator used in the algorithm is non-sequencial four exchange [4,20] using a random walk on the candidate set to find edges to be included in the tour. This operator has been proven to be very effective in conjunction with Lin-Kernighan local search [14]. Hence, in each mutation as few as four edges are exchanged: Edges (t_1, t_2), (t_3, t_4), (t_5, t_6), and (t_7, t_8) are replaced by edges (t_1, t_4), (t_2, t_3), (t_5, t_8), and (t_6, t_7). The random walk omn the candidate edge set assures that edges with a relatively small length instead of arbitrarily long edges are included. We experimented with several other mutation schemes. This one appeared to be the best. We found that a reasonable number of steps for the random walk is 150 independently of the problem size. So, we used this value in our algorithm.

Local Search Operator. As mentioned before, we use a variant of the original Lin-Kernighan heuristic for the local search. Compared to the original LK, we use 3-opt moves as submoves instead of 2-opt moves at all levels: edges (t_1, t_2),

```
function ILK−PM(Instance : TspInstance, MaxIter : Integer) : TspTour;
  begin
    C := FindInitialCandidateSet(Instance);
    Tour := Init ();
    Tour := LocalSearch(C, Tour);
    C := FindInitialLowerBound(C, TourLength(Tour));
    for iter := 1 to MaxIter do begin
      Tbest := Tour;
      Tour := Mutate(Tour);
      Tour := LocalSearch(Tour);
      if TourLength(Tour) < TourLength(Tbest) then Tbest := Tour;
      if (iter % 400) = 0 then C := UpdateLowerBound(C, TourLength(Tbest));
    end
    return Tbest;
  end
```

Fig. 1. The Evolutionary Local Search Algorithm

(t_3, t_4), and (t_5, t_6) are replaced by edges (t_2, t_3), (t_4, t_5), and (t_6, t_1) in a sub move. The next sub move will start by replacing the last new edge (t_6, t_1). We do not use backtracking which simplifies the implementation drastically without affecting the performance. In this aspect our implementation is similar to LKH. As in other LK implentations we make use of Bentley's don't look bits concept [21]. Moreover, we use two-level trees to represent tours [22].

3.3 The Lower Bound Computation

Held and Karp [23,24] proposed a method based on Lagrangian relaxiation to compute lower bounds for the TSP. It is based on computing 1-trees, i.e. minimum spanning trees with one additional edge (the second shortest edge of a leaf in the tree). The approach is to find a transformation given by a vector $\pi = (\pi_1, \ldots, \pi_N)$ that maximizes the lower bound w

$$w(\pi) = L(T_\pi) - 2\sum_{i=1}^{N} \pi_i, \tag{1}$$

where N is the problem size, T_π is a minimum 1-tree on the transformed graph G' for which the cost of traveling between city i and j is $c'_{ij} = c_{ij} + \pi_i + \pi_j$,

```
function UpdateLowerBound(C : CandidateSet; upper : REAL) :
TspTour;
  begin
    if (FirstTime) then begin
      InitPiValues(Pi);
      best_lower : = Calculate1Tree(Pi);
    end
    FirstTime := false;
    t := (upper − best_lower) / norm;
    for i := 1 to 200 do begin
      updatePi(Pi, t);
      lower : = Calculate1Tree(Pi);
      if (lower > best_lower) then begin
        best_lower := lower;
        Best_Pi := Pi;
        t := t ∗ 4.0;
      end
      t := t ∗ 0.75;
    end
    best_lower = Calculate1Tree(Best_Pi);
    return best_lower;
  end
```

Fig. 2. The Incremental Lower Bound Improvement

and $L(T_\pi)$ is the cost of the tree with respect to G'. Compared to other lower bounds this bound does not require to compute a linear program. In order to find the best π vector, a subgradient optimization can be applied. Within the subgradient optimization, π is updated as follows:

$$\pi_i = \pi_i + t^{(k)} \left(0.7(d_i^{(k)} - 2) + 0.3(d_i^{(k-1)} - 2)\right), \tag{2}$$

where $d_i^{(k)}$ denotes the degree of city i in the minimum 1-tree at step k. For $t^{(k)} \rightarrow 0$, for $k \rightarrow \infty$, and $\sum t^{(k)} = \infty$, $w(\pi)$ will converge to the maximum of $w(\pi)$.

However, in practice convergence can be very slow and since computing minimum 1-trees is expensive for very large instances, we use a simplified scheme of adjusting $t^{(k)}$ as shown in Fig. 2 and we repeat the subgradient optimization several times. In the figure, *norm* denotes $\sum_i (d_i^{(k)} - 2)^2$.

Since computing the minimum 1-tree for very large instances is time consuming, we calculate the minimum 1-tree in the candidate set. The final 1-tree calculation of each call to UpdateLowerBound() is computed after the candidate set was recomputed on the transformed instance using the best π vector. The ILK will use the recomputed candidate set in its subsequent iterations.

4 Experimental Evaluation

To assess the performance of our algorithm we performed several runs on a set of publicly available benchmark instances. We used all seven Euclidean TSPLIB instances of size >10,000, three national TSP instances of size >10,000, three VLSI instances of size >100,000 from `http://www.tsp.gatech.edu/`, and finally seven random Euklidean instances form the DIMACS TSP challenge in the range between 10 thousand and 10 million. We report average values of 32 runs for each instance. The algorithms were coded in C++ (under Linux with gcc) and running times are reported for an Intel Core Quad 6600 processor. For each run only one CPU core was used. Results are reported in Table 1. For each instance, the name (containing the size of the instance), the average final tour length, the standard deviation of the tour length (sdev), the percentage excess over the best–known solution (or in case of the E*.0 instances over the Held-Karp lower bound), the gap of the computed lower and upper bound, and the running time in seconds are provided. The termination criterion was $0.1N$ iterations (N is the problem size). The last two columns contain the average percentage excess and the running time for the alogrithm without lower bound computation.

The results demonstrate that the algorithm is capable of finding provably good solutions in very short time: For the TSPLIB instances, the gap lies between 0.64% and 1.51% by spending at most 40 seconds of CPU time. For the national instances, the gap is about 1.0% with a maximum of 122 seconds. The VLSI instances are significantly greater and the running time increases up to 1,192 seconds. The average gap is about 1.5%. Finally, for the random Euclidean instances the gap is below 0.83% independently of the size of the instance. The running time increases from 13 seconds (10 thousand cities) to 32,789 seconds

Table 1. Results of our ILK (ILK-PM-.1N) for DIMACS TSP Challenge Instances

	With Lower Bound					Without LB	
Instance	Tour length	sdev	Excess	Gap	sec	Excess	sec
rl11849	926071.6	583.5	0.302%	1.51%	9	0.382%	4
usa13509	20011164.1	6114.4	0.142%	0.91%	24	0.156%	18
brd14051	470051.6	230.9	0.148%	0.92%	20	0.140%	12
d15112	1574915.0	175.8	0.123%	0.67%	24	0.123%	16
d18512	646089.9	89.5	0.138%	0.64%	22	0.138%	13
pla33810	66456993.5	43691.9	0.618%	1.44%	20	0.693%	6
pla85900	143254481.1	85239.3	0.612%	1.24%	40	0.634%	13
sw24978	857594.8	257.0	0.234%	1.13%	33	0.262%	16
bm33708	961536.8	209.0	0.234%	0.98%	54	0.253%	17
ch71009	4573501.2	547.5	0.153%	0.86%	122	0.163%	70
sra104815	252310.2	47.7	0.377%	1.11%	152	0.443%	50
ara238025	581381.4	157.8	0.435%	1.32%	495	0.511%	121
lra498378	2183574.5	1389.0	0.705%	1.99%	722	0.886%	244
lrb744710	1619145.0	538.5	0.435%	1.36%	1192	0.541%	321
E10k.0	71969032.4	20938.0	0.850%	0.86%	13	0.853%	10
E31k.0	127469229.7	16091.7	0.786%	0.80%	50	0.792%	34
E100k.0	226146457.7	15767.1	0.809%	0.82%	184	0.819%	123
E316k.0	401954702.2	15174.6	0.801%	0.82%	665	0.808%	428
E1M.0	714351765.1	18622.6	0.797%	0.81%	2677	0.804%	1449
E3M.0	1269419125.3	14147.1	0.748%	0.81%	11547	0.755%	5228
E10M.0	2256845968.6	12030.8	0.752%	0.81%	32789	0.760%	17867

(10 million cities). Without the lower bound computation and the update of the candidate set the running time is considerably lower for the larger instances. Moreover, the average final tour quality is in all cases but one lower with candidate set update deactivated. The results are up to 0.18% better for the algorithm with lower bound computation. However, this appears to be dependent on the problem instance. For the uniform random instances, the gain is only about 0.01%. The runtime increases almost linearly with the problem size for our ILK as Fig. 3 demonstrates. In order to compare with other state-of-the-art approaches, Table 2 shows a comparison with the eleven best performing algorithms (out of 90) listed on the DIMACS TSP challenge web page. The summary was produced with the statistics code from the challenge. Thus the running time reported in the table is normalized to a DEC Alpha processor with 500 MHz in order to allow a comparison of the different approaches. The quality is given as the percentage excess over the Held-Karp (HK) bound. As shown in the table, our algorithm provides a significantly better tour quality than the other approaches. And it does this in a fraction of time of the second best approach which is also an ILK implementation. Note that none of the competitors

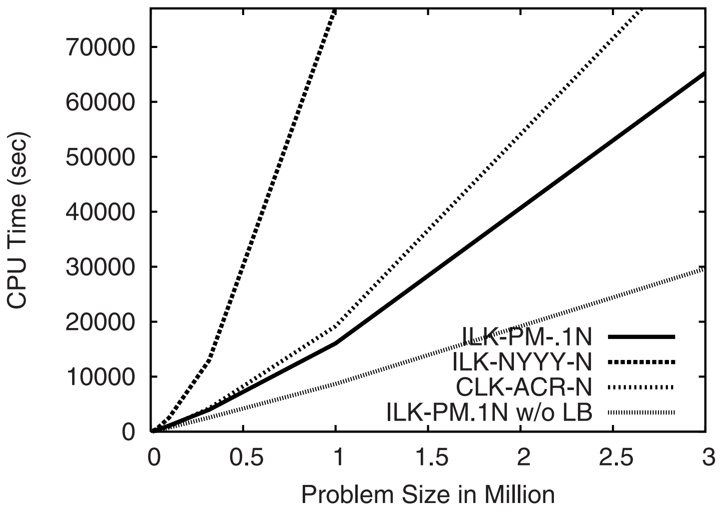


Fig. 3. Scaling behaviour of three ILK variants

computes a lower bound. For the 10 million city instance E10M.0, the quality of our approach is 0.75% over the Held-Karp bound compared to the best algorithm of the DIMACS challenge which is 1.63% over the Held-Karp bound!

Table 2. Comparison of DIMACS TSP Challenge Results on E1M.0. ILK-PM-.1N denotes our ILK with 1 million iterations and ILK-PM-.1N denotes our ILK with 1,2 million iterations.

% HK	Seconds	Implementation	Reference
0.787	17544.0	ILK-PM-.12N	this paper
0.792	77161.6	ILK-NYYY-N	([25])
0.797	16062.0	ILK-PM-.1N	this paper
0.804	8694.0	ILK-PM-.1N without LB	this paper
0.841	6334.0	ILK-NYYY-Ng	([25])
0.879	42242.5	MLCLK-N	[10]
0.888	3480.2	ILK-NYYY-.5Ng	([25])
0.903	19182.7	BSDP-6	[15]
0.903	19503.1	BSDP-8	[15]
0.903	21358.3	BSDP-10	[15]
0.903	19108.1	CLK-ABCC-N.Sparc	[13]
0.905	19192.3	CLK-ACR-N	[14]
0.910	16008.0	CLK-ABCC-N.MIPS	[13]
0.945	20907.6	MLCLK-.5N	[10]

This is due to the fact that the best algorithms for the smaller instances do not scale as well as our approach: Fig. 3 shows the normalized runtime of our approach (ILK-PM-.1 with and without lower bound computation), the ILK of Nguyen *et al.* (for which no reference exists except for the DIMACS challenge web page) denoted ILK-NYYY-N, and the runtime of chained LK of Applegate *et al.* denoted CLK-ACR-N [14] depending on the problem size. While the runtime of our approach without lower bound computation grows linearly with the problem size, the runtime of the others clearly grows faster and and yields in the non-applicability of these algorithms to very large problem instances (>1 million) whereas our approach is still very successful even if the lower bound computation is activated.

5 Conclusions

We presented a new Iterated Lin-Kernighan heuristic based on the powerful concept of iterated local search. We have shown in experiments that our approach scales well with the problem size and is therefore applicable to very large TSP instances with 10 million cities. Besides the scalability, the approach provides provably good solutions since it computes a lower bound interleaved with the optimization. Therefore, the obtained results are known to be not more than about 1% above the optimum and in case of the very large random Euclidean instances not more than 0.81% above the Held-Karp Lower bound even for the largest, 10 million cities instances. Compared to other evolutionary and non-evolutionary approaches for very large instances above 1 million cities, our approach obtains better tour quality in even shorter time. In particular, all algorithms from the DIMACS TSP implementation challenge are shown to be inferior to our approach. To the best of our knowledge the proposed approach is the only one that is both scalable to millions of cities and provides provably good solutions.

There are some issues for future research. Currently, we are working on a distributed algorithm for large instances based on the ILK presented here. Both the use of a population and the use of recombination are subject of our studies. Finally, we believe that our lower bound computation can be further improved.

References

1. Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a Large-Scale Traveling Salesman Problem. Operations Research 2, 393–410 (1954)
2. Arora, S.: Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems. Journal of the ACM 45, 753–782 (1998)
3. Applegate, D., Bixby, R., Chvátal, V., Cook, B.: Finding Cuts in the TSP (A preliminary report). Technical Report 95-05, DIMACS (1995)
4. Lin, S., Kernighan, B.: An Effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Research 21, 498–516 (1973)

5. Helsgaun, K.: An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal of Operational Research 126, 106–130 (2000)
6. Nagata, Y.: New EAX Crossover for Large TSP Instances. In: Runarsson, T.P., Beyer, H.G., Burke, E., Merelo-Guervos, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 372–381. Springer, Heidelberg (2006)
7. Nguyen, H.D., Yoshihara, I., Yamamori, K., Yasunaga, M.: Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems. IEEE Transactions on Systems, Man and Cybernetics, Part B 37, 92–99 (2007)
8. Mühlenbein, H.: Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. In: Schaffer, J.D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 416–421. Morgan Kaufmann, San Francisco (1989)
9. Johnson, D.S., McGeoch, L.A.: Experimental Analysis of Heuristics for the STSP. In: Gutin, G., Punnen, A. (eds.) The Traveling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht (2002)
10. Walshaw, C.: A Multilevel Approach to the Travelling Salesman Problem. Operations Research 50, 862–877 (2002)
11. Lourenco, H.R., Martin, O., Stützle, T.: Iterated Local Search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, Kluwer Academic Publishers, Dordrecht (2003)
12. Johnson, D.S.: Local Optimization and the Traveling Salesman Problem. In: Paterson, M. (ed.) ICALP 1990. LNCS, vol. 443, pp. 446–461. Springer, Heidelberg (1990)
13. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Finding Tours in the TSP. Technical Report Report Number 99885, Research Institute for Discrete Mathematics, University of Bonn, Germany (1999)
14. Applegate, D., Cook, W., Rohe, A.: Chained Lin-Kernighan for Large Traveling Salesman Problems. INFORMS Journal on Computing 15, 82–92 (2003)
15. Balas, E., Simonetti, N.: Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs: A Computational Study. INFORMS J. on Computing 13, 56–75 (2000)
16. Marinakis, Y., Migdalas, A., Pardalos, P.M.: A Hybrid Genetic–GRASP Algorithm Using Lagrangean Relaxation for the Traveling Salesman Problem. Journal of Combinatorial Optimization 10, 311–326 (2005)
17. Merz, P., Freisleben, B.: Fitness Landscapes and Memetic Algorithm Design. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 245–260. McGraw–Hill, London (1999)
18. Merz, P.: Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany (2000)
19. Bentley, J.L.: K-d-Trees for Semidynamic Point Sets. In: Proceedings of the Sixth Annual ACM Symposium on Computational Geometry, pp. 187–197 (1990)
20. Merz, P., Freisleben, B.: Memetic Algorithms for the Traveling Salesman Problem. Complex Systems 13, 297–345 (2001)
21. Bentley, J.L.: Experiments on Traveling Salesman Heuristics. In: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 91–99 (1990)
22. Fredman, M.L., Johnson, D.S., McGeoch, L.A., Ostheimer, G.: Data Structures for Traveling Salesmen. Journal of Algorithms 18, 432–479 (1995)

23. Held, M., Karp, R.M.: The Traveling-Salesman Problem and Minimum Spanning Trees. Operations Research 18, 1138–1162 (1970)
24. Held, M., Karp, R.M.: The Traveling-Salesman Problem and Minimum Spanning Trees: Part II. Mathematical Programming 1, 6–25 (1971)
25. Nguyen, H.D., Yoshihara, I., Yamamori, K., Yasunaga, M.: A New Three-Level Tree Data Structure for Representing TSP Tours in the Lin-Kernighan Heuristic. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences E90-A, 2187–2193 (2007)

Intrinsic System Model of the Genetic Algorithm with α-Selection

André Neubauer

Information Processing Systems Lab
Münster University of Applied Sciences
Stegerwaldstraße 39, D-48565 Steinfurt, Germany
andre.neubauer@fh-muenster.de

Abstract. Genetic algorithms are random heuristic search (RHS) algorithms which can be theoretically described with the help of a dynamical system model. This model characterises the stochastic trajectory of a population using a deterministic heuristic function and its fixed points. For practical problem sizes the determination of the fixed points is unfeasible even for the simple genetic algorithm (SGA). In this paper the novel intrinsic system model is introduced for the genetic algorithm with α-selection and the corresponding unique fixed point is determined. It is shown that this model is compatible with the equivalence relation imposed by schemata. In addition to the theoretical analysis experimental results are presented which confirm the theoretical predictions.

1 Introduction

Genetic algorithms mimic biological evolution and molecular genetics in simplified form [1]. These random heuristic search (RHS) algorithms are based on populations of individuals which evolve according to selection and genetic operators like crossover and mutation. They can be theoretically described with the help of a dynamical system model. In this model the algorithm's stochastic dynamics is modeled with an underlying deterministic heuristic function which describes the expected next population [3,6,7]. The population trajectory is attracted by the fixed points of the heuristic function. However, even for moderate problem sizes the determination of the fixed points becomes unfeasible.

The genetic algorithm with α-selection recently introduced in [2] allows to explicitly determine the fixed points of the heuristic function. Due to the selection operation the dynamical system model of genetic algorithms in general is not compatible with the equivalence relation imposed by schemata [4,6]. This paper presents a novel *intrinsic system model* which yields a compatible model and further simplifies the mathematical analysis in [2]. The intrinsic system model also provides a means to analyse the genetic algorithm's exploitation and exploration of the search space irrespective of the fitness function.

The paper is organised as follows. The genetic algorithm with α-selection is described in Sect. 2 based on the notion of the best individual randomly mating with other individuals in the current population. In Sect. 3 the corresponding dynamical system model is derived. The intrinsic system model is introduced in Sect. 4. Its unique fixed point is derived and its compatibility to schemata is proven in Sect. 5. Simulation results presented in Sect. 6 show close agreement between theory and experiment. A brief conclusion is given in Sect. 7.

2 Genetic Algorithm with α-Selection

In this section the genetic algorithm with α-selection, 1-point crossover and bitwise mutation is described following the notation and definition of the simple genetic algorithm (SGA) in [6]. It is assumed that the genetic algorithm is used for the maximisation of a fitness function $f : \Omega \to \mathbb{R}$ which is defined over the search space $\Omega = \mathbb{Z}_2^\ell = \{0,1\}^\ell$ consisting of binary ℓ-tuples $(a_0, a_1, \ldots, a_{\ell-1})$.

Each binary ℓ-tuple $(a_0, a_1, \ldots, a_{\ell-1}) = a_0 a_1 \ldots a_{\ell-1}$ will be identified with the integer $a = a_0 \cdot 2^{\ell-1} + a_1 \cdot 2^{\ell-2} + \ldots + a_{\ell-1} \cdot 2^0$ leading to the search space $\Omega = \{0, 1, \ldots, n-1\}$ with cardinality $|\Omega| = n = 2^\ell$. The fitness values are given by $f(a) = f_a$. Based on the binary number representation the bitwise modulo-2 addition $a \oplus b$, bitwise modulo-2 multiplication $a \otimes b$ and bitwise binary complement $\overline{a}$ are defined. Vice versa, the integer $a \in \Omega$ is viewed as a column vector $(a_0, a_1, \ldots, a_{\ell-1})^\mathrm{T}$. The all-one ℓ-tuple $\mathbf{1}$ corresponds to the integer $n-1 = 2^\ell - 1$. The indicator function is defined by $[i = j] = 1$ if $i = j$ and 0 if $i \neq j$.

Similar to the SGA the genetic algorithm with α-selection works over populations $P(t)$ defined as multisets of r individual binary ℓ-tuples $a(t) \in \Omega$. For the creation of offspring individuals in each generation t genetic operators like crossover χ_Ω and mutation μ_Ω are applied to parental individuals (see Fig. 1).

```
$t := 0$;
initialise population $P(0)$;
**while** end of adaptation $\neq$ true **do**
    select $\alpha$-individual $b(t)$ as first parent;
    **for** the creation of $r$ offspring **do**
        select second parent $c(t)$ randomly;
        create offspring $a(t+1) := \mu_\Omega(\chi_\Omega(b(t), c(t)))$;
    **end**
    increment $t := t + 1$;
**end**
```

Fig. 1. Genetic algorithm with α-selection [2]

For the *α-selection* scheme let $b = \operatorname{argmax}\{f(i) : i \in P\}$ be the best individual or *α-individual* in the current population P. In the genetic algorithm with α-selection the α-individual b is mated with individuals randomly chosen from the current population P with uniform probability r^{-1}.

The *crossover* operator $\chi_\Omega : \Omega \times \Omega \to \Omega$ randomly generates an offspring ℓ-tuple $a = (a_0, a_1, \ldots, a_{\ell-1})$ according to $a = \chi_\Omega(b, c)$ with crossover probability χ from two ℓ-tuples $b = (b_0, b_1, \ldots, b_{\ell-1})$ and $c = (c_0, c_1, \ldots, c_{\ell-1})$. With the crossover mask $m \in \Omega$ the ℓ-tuples $a = b \otimes m \oplus \overline{m} \otimes c$ or $a = b \otimes \overline{m} \oplus m \otimes c$ are generated one of which is chosen as offspring a with equal probability 2^{-1}. For 1-point crossover the crossover mask m is randomly chosen from Ω according to the probability distribution vector $\boldsymbol{\chi} = (\chi_0, \chi_1, \ldots, \chi_{n-1})^{\mathrm{T}}$ with [6]

$$\chi_m = \begin{cases} 1 - \chi \ , & m = 0 \\ \dfrac{\chi}{\ell - 1} \ , & m = 2^\lambda - 1 \text{ with } 1 \leq \lambda \leq \ell - 1 \\ 0 \ , & \text{otherwise} \end{cases} \quad .$$

The bitwise *mutation* operator $\mu_\Omega : \Omega \to \Omega$, which randomly flips each bit of the ℓ-tuple $a = (a_0, a_1, \ldots, a_{\ell-1})$ with mutation probability μ, is defined with the help of the mutation mask $m \in \Omega$ according to $\mu_\Omega(a) = a \oplus m$. The mutation mask m is randomly chosen from Ω according to the probability distribution vector $\boldsymbol{\mu} = (\mu_0, \mu_1, \ldots, \mu_{n-1})^{\mathrm{T}}$ with [6]

$$\mu_m = \mu^{\mathbf{1}^{\mathrm{T}} m} \cdot (1 - \mu)^{\ell - \mathbf{1}^{\mathrm{T}} m} \quad .$$

3 Dynamical System Model

Following [6] the dynamical system model for the genetic algorithm with α-selection will be derived in this section. The dynamics of the genetic algorithm is compactly formulated by defining the population vector $\boldsymbol{p} = (p_0, p_1, \ldots, p_{n-1})^{\mathrm{T}}$. Each component $p_i = r^{-1} \sum_{j \in P} [j = i]$ gives the proportion of element $i \in \Omega$ in the current population P. The population vector $\boldsymbol{p}$ is an element of the simplex $\Lambda = \left\{ \boldsymbol{p} \in \mathrm{I\!R}^n : p_i \geq 0 \wedge \sum_{i \in \Omega} p_i = 1 \right\}$. In the limit of infinite populations with $r \to \infty$ the population vectors are dense in the simplex Λ. With the help of the population vector $\boldsymbol{p}$ the best individual in the current population is defined by

$$b = \operatorname{argmax} \{ f_i : i \in \Omega \wedge p_i > 0 \} \quad .$$

The genetic algorithm with α-selection can be described as an instance of RHS $\tau : \Lambda \to \Lambda$ according to $\boldsymbol{p}(t+1) = \tau(\boldsymbol{p}(t))$. The RHS τ can be equivalently represented by a heuristic function $\mathcal{G} : \Lambda \to \Lambda$ which for a given population vector $\boldsymbol{p}$ yields the probability distribution $\mathcal{G}(\boldsymbol{p})$. This probability distribution $\mathcal{G}(\boldsymbol{p})_i = \Pr\{\text{individual } i \text{ is sampled from } \Omega\}$ is used to generate the next population. The stochastic trajectory $\boldsymbol{p}$, $\tau(\boldsymbol{p})$, $\tau^2(\boldsymbol{p})$, $\ldots$ approximately follows the trajectory $\boldsymbol{p}$, $\mathcal{G}(\boldsymbol{p})$, $\mathcal{G}^2(\boldsymbol{p})$, $\ldots$ of the deterministic dynamical system defined by $\mathcal{G}$. The RHS τ behaves like the dynamical system model in the limit of infinite populations. τ shows punctuated equilibria, i.e. phases of relative stability nearby a fixed point $\boldsymbol{\omega} = \mathcal{G}(\boldsymbol{\omega})$ of the heuristic function $\mathcal{G}$ disrupted by sudden transitions to another dynamical equilibrium near another fixed point. In the following this observation will be referred to as the *fixed point hypothesis* of genetic algorithms.

3.1 Heuristic

In a genetic algorithm with α-selection the α-individual b is selected as the first parent for creation of a new offspring, whereas the second parent is chosen uniformly at random from the current population according to the probability distribution p_j over Ω with $j \in \Omega$. The heuristic function $\mathcal{G}(\boldsymbol{p})$ follows to

$$\mathcal{G}(\boldsymbol{p})_i = \sum_{j \in \Omega} p_j \cdot \Pr\{\mu_\Omega(\chi_\Omega(b,j)) = i\} \ .$$

The probability distributions for crossover χ_Ω and mutation μ_Ω lead to

$$\Pr\{\mu_\Omega(\chi_\Omega(b,j)) = i\} = \sum_{u,v \in \Omega} \mu_v \cdot \frac{\chi_u + \chi_{\overline{u}}}{2} \cdot [b \otimes u \oplus \overline{u} \otimes j = i \oplus v] \ .$$

By defining the $n \times n$ *mixing matrix* [6]

$$M_{i,j} = \sum_{u,v \in \Omega} \mu_v \cdot \frac{\chi_u + \chi_{\overline{u}}}{2} \cdot [i \otimes u \oplus \overline{u} \otimes j = v] \tag{1}$$

this yields $\Pr\{\mu_\Omega(\chi_\Omega(b,j)) = i\} = M_{i \oplus b, i \oplus j}$ and finally

$$\mathcal{G}(\boldsymbol{p})_i = \sum_{j \in \Omega} p_j \cdot M_{i \oplus b, i \oplus j} \ .$$

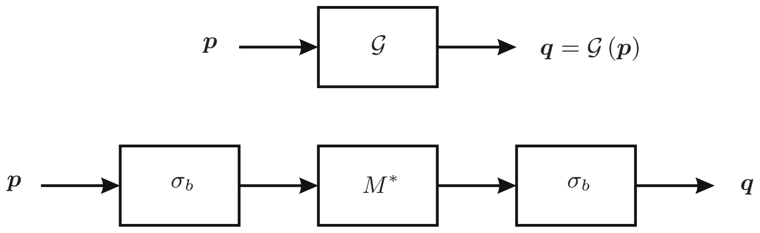
With the permutation matrix $(\sigma_b)_{i,j} = [i \oplus j = b]$ and the *twist* $(M^*)_{i,j} = M_{i \oplus j, i}$ of the symmetric mixing matrix $M = M^{\mathrm{T}}$ the new population vector is given by

$$\boxed{\boldsymbol{q} = \mathcal{G}(\boldsymbol{p}) = \sigma_b \cdot M^* \cdot \sigma_b \cdot \boldsymbol{p}} \ . \tag{2}$$

This *dynamical system model* is illustrated in Fig. 2.

3.2 Mixing Matrix

The calculation of the mixing matrix M can be done efficiently with the help of the Walsh transform [5]. For a matrix M the Walsh transform is $\widehat{M} = W \cdot M \cdot W$ with the symmetric and orthogonal $n \times n$ Walsh matrix $W_{i,j} = n^{-1/2} \cdot (-1)^{i^{\mathrm{T}} j}$.


$$\begin{tikzcd} \boldsymbol{p} \arrow[r] & \boxed{\mathcal{G}} \arrow[r] & \boldsymbol{q} = \mathcal{G}(\boldsymbol{p}) \end{tikzcd}$$

$$\begin{tikzcd} \boldsymbol{p} \arrow[r] & \boxed{\sigma_b} \arrow[r] & \boxed{M^*} \arrow[r] & \boxed{\sigma_b} \arrow[r] & \boldsymbol{q} \end{tikzcd}$$

Fig. 2. Dynamical system model of the genetic algorithm with α-selection with heuristic function $\mathcal{G}$ (top) defined by the matrix $\sigma_b \cdot M^* \cdot \sigma_b$ (bottom)

The WALSH transform of a vector $\boldsymbol{v}$ yields $\widehat{\boldsymbol{v}} = W \cdot \boldsymbol{v}$. For 1-point crossover χ_Ω and bitwise mutation μ_Ω the WALSH transform of the mixing matrix M is [6]

$$\widehat{M}_{i,j} = [i \otimes j = 0] \cdot (1 - 2\mu)^{\mathbf{1}^{\mathrm{T}}(i \oplus j)} \cdot \left((1 - \chi) \cdot \frac{[i = 0] + [j = 0]}{2} + \chi \cdot \frac{(\mathrm{lo}\,(j) - \mathrm{hi}\,(i))^+ + (\mathrm{lo}\,(i) - \mathrm{hi}\,(j))^+}{2 \cdot (\ell - 1)} \right) .$$

Here, $\mathrm{hi}\,(k) = 0$ and $\mathrm{lo}\,(k) = \ell - 1$ if $k = 0$ as well as $\mathrm{hi}\,(k) = \sup \left\{ \lambda : 2^\lambda \otimes k > 0 \right\}$ and $\mathrm{lo}\,(k) = \inf \left\{ \lambda : 2^\lambda \otimes k > 0 \right\}$ if $k \neq 0$; $(\cdot)^+$ denotes the maximum of its argument and 0. The WALSH transform of the twist of the mixing matrix can be calculated from

$$\left(M^{*\wedge} \right)_{i,j} = \widehat{M}_{i \oplus j, j} \quad . \tag{3}$$

4 Intrinsic System Model

The matrix $\sigma_b \cdot M^* \cdot \sigma_b$ of the dynamical system model $\boldsymbol{q} = \sigma_b \cdot M^* \cdot \sigma_b \cdot \boldsymbol{p}$ in (2) of the genetic algorithm with α-selection depends on the mixing matrix M and the α-individual b. Because of $\sigma_b^{-1} = \sigma_b$ this yields the equivalent formulation

$$\boxed{\sigma_b\, \boldsymbol{q} = M^* \cdot \sigma_b\, \boldsymbol{p}} \quad . \tag{4}$$

The permuted population vector $\sigma_b\, \boldsymbol{p}$ develops according to the matrix M^* which is independent of the α-individual b, i.e. the diagram in Fig. 3 commutes.

M^* defines the *intrinsic system model* of the genetic algorithm with α-selection. It corresponds to the dynamical system model with $b = 0$. Because of the corresponding linear equation $\boldsymbol{q} = M^* \cdot \boldsymbol{p}$ the fixed points of the intrinsic system model are obtained from the eigenvectors of M^* to eigenvalue $\lambda = 1$, i.e. $M^* \cdot \boldsymbol{\omega} = \boldsymbol{\omega}$. The fixed points of the heuristic function $\mathcal{G}$ of the dynamical system model follow from the permutation $\sigma_b\, \boldsymbol{\omega}$ for a given α-individual b. For the fixed point analysis of the dynamical system model it therefore suffices to analyse the intrinsic system model shown in Fig. 4. To this end the WALSH transform of both sides of the equation $\boldsymbol{q} = M^* \cdot \boldsymbol{p}$ is taken yielding $\widehat{\boldsymbol{q}} = M^{*\wedge} \cdot \widehat{\boldsymbol{p}}$. For an eigenvector $\boldsymbol{v}$ with eigenvalue λ it follows $M^* \cdot \boldsymbol{v} = \lambda \cdot \boldsymbol{v}$ and equivalently $M^{*\wedge} \cdot \widehat{\boldsymbol{v}} = \lambda \cdot \widehat{\boldsymbol{v}}$, i.e. the matrix M^* and its WALSH transform $M^{*\wedge}$ have the same eigenvalues.

$$\begin{tikzcd}
\boldsymbol{p} \arrow[r, "\mathcal{G}"] \arrow[d, "\sigma_b"'] & \mathcal{G}\,(\boldsymbol{p}) \arrow[d, "\sigma_b"] \\
\sigma_b\, \boldsymbol{p} \arrow[r, dashed, "M^*"'] & \sigma_b\, \mathcal{G}\,(\boldsymbol{p})
\end{tikzcd}$$

Fig. 3. Commutativity diagram for the dynamical system model of the genetic algorithm with α-selection with heuristic function $\mathcal{G}$ and permutation σ_b

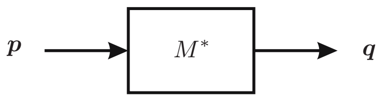


Fig. 4. Intrinsic system model of the genetic algorithm with α-selection

For crossover and mutation the WALSH transform of the mixing matrix fulfills $\widehat{M}_{i,j} \propto [i \otimes j = 0]$, i.e. $\widehat{M}$ is separative. $M^{*\wedge} = M^{\wedge **}$ is a lower triangular matrix the spectrum of which is given by the first column of $\widehat{M}$ [6]. Since the spectrum of M^* and its WALSH transform $M^{*\wedge}$ are the same this yields the eigenvalues

$$\lambda_i = \left(M^{*\wedge}\right)_{i,i} = \widehat{M}_{0,i} \ .$$

Because of $\lambda_0 = 1$ and $0 \leq \lambda_i \leq \frac{1}{2} - \mu < \frac{1}{2}$ for $1 \leq i \leq n-1$ there exists a single eigenvector $\boldsymbol{\omega}$ which is a unique fixed point of the intrinsic system model

$$\boxed{\boldsymbol{\omega} = M^* \cdot \boldsymbol{\omega}} \ . \tag{5}$$

The fixed points of the heuristic function $\mathcal{G}$ of the genetic algorithm with α-selection are obtained from the permutation $\sigma_b \, \boldsymbol{\omega}$ for a given α-individual b. According to the *fixed point hypothesis* the population will stay near this fixed point $\sigma_b \, \boldsymbol{\omega}$ and converge to a new fixed point if a better α-individual is found.

The unique fixed point $\boldsymbol{\omega}$ of the intrinsic system model can be determined explicitly with the help of the WALSH transform. Due to the relation $\widehat{\boldsymbol{\omega}} = M^{*\wedge} \cdot \widehat{\boldsymbol{\omega}}$ and the lower triangular matrix $M^{*\wedge}$ the WALSH transform of the fixed point can be recursively calculated according to

$$\boxed{\widehat{\omega}_i = \frac{1}{1 - \widehat{M}_{0,i}} \cdot \sum_{j=0}^{i-1} \widehat{M}_{i \oplus j, j} \cdot \widehat{\omega}_j} \tag{6}$$

for $1 \leq i \leq n-1$ starting with $\widehat{\omega}_0 = n^{-1/2}$ which ensures $\sum_{i \in \Omega} \omega_i = 1$. The fixed point is then obtained via the inverse WALSH transform $\boldsymbol{\omega} = W \cdot \widehat{\boldsymbol{\omega}}$.

5 Schemata

In this section coarse-grained system models based on schemata will be explored as equivalence relations [6]. Two equivalent individuals $i \equiv j$ in the search space Ω belong to the same equivalence class $[i] = \{j \in \Omega : j \equiv i\}$. This can be expressed with the help of the *quotient map* $\Xi_{[i],j} = [i \equiv j]$, i.e. $i \equiv j$ if $\Xi_{[i],j} = 1$. Two populations are equivalent if the proportions of individuals in each of the equivalence classes $[i]$ with $i \in \Omega$ are the same in both populations. By using the population vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ in the simplex Λ this corresponds to $\Xi \boldsymbol{p} = \Xi \boldsymbol{q}$.

According to [6] *schemata* can be considered as specific equivalence relations. A *schemata family* is defined with the help of the ℓ-tuple $\xi \in \Omega$ via

$\Xi_{[i],j} = [j \otimes \xi = i]$ leading to the $2^{\mathbf{1}^{\mathrm{T}}\xi} \times 2^{\ell}$ matrix Ξ. Here, $i \in \Omega_\xi = \{i \in \Omega : i \otimes \overline{\xi} = 0\}$ and $j \in \Omega$. Two individuals $j, k \in \Omega$ are equivalent if they agree on the defining positions according to $j \equiv k \Leftrightarrow j \otimes \xi = k \otimes \xi$. The number of the defining positions is $\mathbf{1}^{\mathrm{T}}\xi$ which yields the cardinality $|\Omega_\xi| = 2^{\mathbf{1}^{\mathrm{T}}\xi}$.

5.1 Schema Heuristic

Based on the intrinsic system model of the genetic algorithm with α-selection

$$q_i = \sum_{j \in \Omega} p_j \cdot M_{i,i \oplus j} \tag{7}$$

a coarse-grained system model will now be derived. The proportion of the expected next population representing *schema* $[i] = i \oplus \Omega_{\overline{\xi}}$ with $i \in \Omega_\xi$ can be calculated according to

$$\widetilde{q}_{[i]} = (\Xi \boldsymbol{q})_{[i]} = \sum_{j \in \Omega} [j \otimes \xi = i] \cdot q_j = \sum_{j \in \Omega_{\overline{\xi}}} q_{i \oplus j} = \sum_{j \in \Omega_{\overline{\xi}}} \sum_{k \in \Omega} p_k \cdot M_{i \oplus j, i \oplus j \oplus k} \ .$$

This yields

$$\widetilde{q}_{[i]} = \sum_{j \in \Omega_\xi} \widetilde{p}_{[j]} \cdot (M_\xi)_{[i],[i \oplus j]} \tag{8}$$

with $\widetilde{\boldsymbol{p}} = \Xi \boldsymbol{p}$ and the symmetric $2^{\mathbf{1}^{\mathrm{T}}\xi} \times 2^{\mathbf{1}^{\mathrm{T}}\xi}$ *schema mixing matrix* [6]

$$(M_\xi)_{[i],[j]} = \sum_{u,v \in \Omega_\xi} (\Xi \mu)_{[v]} \cdot \frac{(\Xi \chi)_{[u]} + (\Xi \chi)_{[\overline{u}]}}{2} \cdot [i \otimes u \oplus \overline{u} \otimes j = v] \ . \tag{9}$$

The coarse-grained system model based on schemata for the intrinsic system model of a genetic algorithm with α-selection is therefore given by

$$\boxed{\widetilde{\boldsymbol{q}} = M_\xi^* \cdot \widetilde{\boldsymbol{p}}} \ . \tag{10}$$

The intrinsic system model is compatible with the equivalence relation defined by the schemata family ξ because the diagram in Fig. 5 commutes. This conforms to the observation that the mixing operation of the SGA with crossover and mutation is compatible with this equivalence relation [6].

5.2 Schema Mixing Matrix

The twist of the schema mixing matrix M_ξ can be expressed with the help of the twist of the mixing matrix M and the quotient map Ξ according to

$$\boxed{M_\xi^* = \frac{2^{\mathbf{1}^{\mathrm{T}}\xi}}{n} \cdot \Xi \cdot M^* \cdot \Xi^{\mathrm{T}}} \ . \tag{11}$$

$$\begin{tikzcd}
\boldsymbol{p} \arrow[r, "M^*"] \arrow[d, "\Xi"'] & \boldsymbol{q} \arrow[d, "\Xi"] \\
\widetilde{\boldsymbol{p}} \arrow[r, dashed, "M^*_\xi"'] & \widetilde{\boldsymbol{q}}
\end{tikzcd}$$

Fig. 5. Commutativity diagram for intrinsic system model M^* with quotient map Ξ

With the $2^{\mathbf{1}^{\mathrm{T}}\xi} \times 2^{\mathbf{1}^{\mathrm{T}}\xi}$ WALSH matrix W_ξ over Ω_ξ and $i, j \in \Omega_\xi$ the WALSH transform $M_\xi^{*\wedge} = W_\xi \cdot M_\xi^* \cdot W_\xi$ follows to

$$(M_\xi^{*\wedge})_{[i],[j]} = (\widehat{M}_\xi)_{[i \oplus j],[j]} \quad . \tag{12}$$

$M_\xi^{*\wedge}$ is obtained from $M^{*\wedge}$ by choosing rows and columns with indices in Ω_ξ, i.e.

$$\boxed{(M_\xi^{*\wedge})_{[i],[j]} = (M^{*\wedge})_{i,j}} \quad . \tag{13}$$

5.3 Schema Fixed Point

The matrix M_ξ^* and its WALSH transform $M_\xi^{*\wedge}$ have the same eigenvalues. Because of (13) for a lower triangular matrix $M^{*\wedge}$ the matrix $M_\xi^{*\wedge}$ is also lower triangular. The corresponding eigenvalues are obtained from

$$\lambda_{[i]} = (M_\xi^{*\wedge})_{[i],[i]} = (M^{*\wedge})_{i,i} = \lambda_i$$

with $i \in \Omega_\xi$, i.e. the eigenvalues $\lambda_{[i]}$ correspond to the eigenvalues λ_i. There exists a single eigenvalue $\lambda_{[0]} = 1$ which leads to the unique schema fixed point

$$\boxed{\widetilde{\boldsymbol{\omega}} = M_\xi^* \cdot \widetilde{\boldsymbol{\omega}}} \quad . \tag{14}$$

The unique schema fixed point $\widetilde{\boldsymbol{\omega}}$ can be determined explicitly from the relation $\widehat{\widetilde{\boldsymbol{\omega}}} = M_\xi^{*\wedge} \cdot \widehat{\widetilde{\boldsymbol{\omega}}}$ by taking into account that $M_\xi^{*\wedge}$ is a lower triangular matrix. The schema fixed point is then obtained via the inverse WALSH transform $\widetilde{\boldsymbol{\omega}} = W_\xi \cdot \widehat{\widetilde{\boldsymbol{\omega}}}$.

6 Experimental Results

In this section the ONEMAX problem with fitness function $f_i = \mathbf{1}^{\mathrm{T}} i$ is considered, i.e. f_i denotes the number of 1's in the binary representation of $i \in \Omega$. A genetic algorithm with α-selection using 1-point crossover, bitwise mutation and random initial population is used with the strategy parameters $\ell = 10$, $n = 1024$, $\chi = 0.5$, $\mu = \ell^{-1}$ and $r = 50$. A schemata family is defined by the binary ℓ-tuple

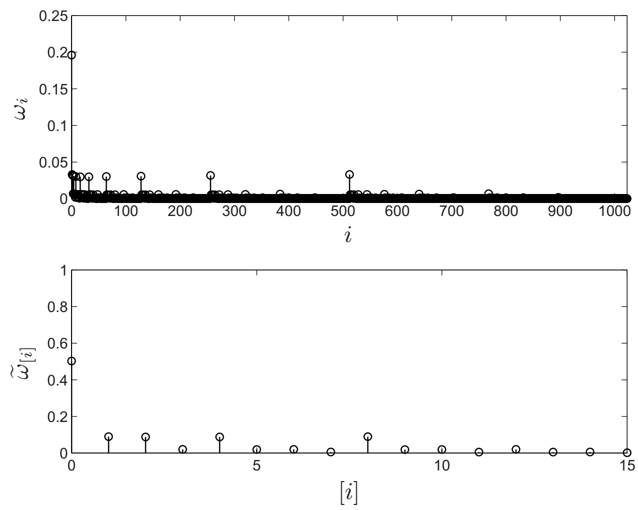


Fig. 6. Fixed points of intrinsic system model (top) and schema heuristic (bottom)

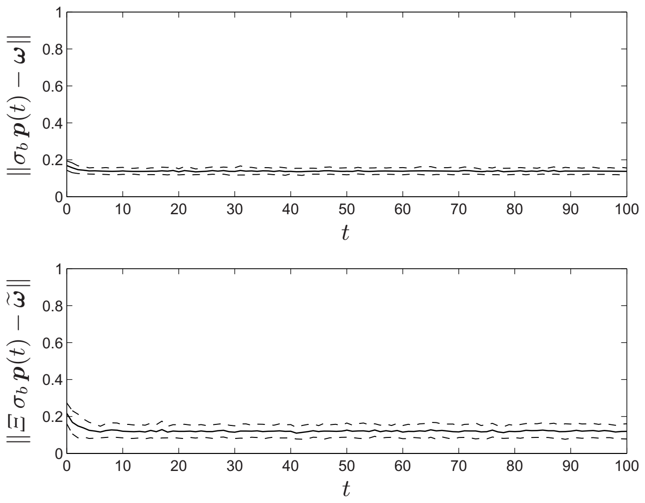


Fig. 7. EUCLIDean distances $\|\sigma_b\, \boldsymbol{p}(t) - \boldsymbol{\omega}\|$ (top) and $\|\Xi\, \sigma_b\, \boldsymbol{p}(t) - \widetilde{\boldsymbol{\omega}}\|$ (bottom) over t

$\xi = 000000\underline{1111}$ or $\xi = 15$, respectively. The fixed points $\boldsymbol{\omega}$ and $\widetilde{\boldsymbol{\omega}}$ of the intrinsic system model and the schema heuristic, respectively, are shown in Fig. 6.

The EUCLIDean distances of the simulated and permuted population vectors $\sigma_b\, \boldsymbol{p}(t)$ or $\Xi\, \sigma_b\, \boldsymbol{p}(t)$ in generation t to the fixed points $\boldsymbol{\omega}$ or $\widetilde{\boldsymbol{\omega}}$, respectively, are

$$\|\sigma_b\, \boldsymbol{p}(t) - \boldsymbol{\omega}\| = \sqrt{\sum\nolimits_{i \in \Omega} \left((\sigma_b\, \boldsymbol{p}(t))_i - \omega_i \right)^2} \ ,$$

$$\|\Xi\,\sigma_b\,\boldsymbol{p}(t) - \widetilde{\boldsymbol{\omega}}\| = \sqrt{\sum\nolimits_{i\in\Omega_\xi}\left((\Xi\,\sigma_b\,\boldsymbol{p}(t))_{[i]} - \widetilde{\omega}_{[i]}\right)^2}\ .$$

In Fig. 7 these EUCLIDean distances are shown averaged over 100 simulation runs also indicating the range of $\pm$ one standard deviation. As these results illustrate there is a close match between theoretical predictions and experimental results.

7 Conclusion

The intrinsic system model for the genetic algorithm with α-selection simplifies the analysis of the dynamical system model of genetic algorithms. It is defined by the mixing matrix M and enables the derivation of the unique fixed point $\boldsymbol{\omega}$. The simplifications are gained because the fitness function f is hidden from the mathematical formulation by making use of the α-individual b. Since b enters the dynamical system model via a permutation σ_b according to $\sigma_b \cdot M^* \cdot \sigma_b$ the intrinsic system model can be formulated with the help of the matrix M^*.

The intrinsic system model provides a means to analyse the genetic algorithm's exploitation and exploration of the search space Ω irrespective of the fitness function f. This model is compatible with the equivalence relation imposed by schemata as was shown by explicitly deriving the coarse-grained system model for a given schemata family ξ. Experimental results showed close agreement to the theoretical predictions obtained from the intrinsic system model.

Acknowledgement

The author would like to thank the anonymous reviewer of [2] for the suggestions leading to the intrinsic system model of the genetic algorithm with α-selection.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems, 1st edn. MIT Press, Cambridge (1992)
2. Neubauer, A.: Theory of the Simple Genetic Algorithm with α-Selection. In: Genetic and Evolutionary Computation Conference GECCO (to appear, 2008)
3. Reeves, C.R., Rowe, J.E.: Genetic Algorithms – Principles and Perspectives, A Guide to GA Theory. Kluwer Academic Publishers, Boston (2003)
4. Rowe, J.E., Vose, M.D., Wright, A.H.: Coarse Graining Selection and Mutation. In: Foundations of Genetic Algorithms. LNCS, pp. 176–191. Springer, Heidelberg (2005)
5. Vose, M.D., Wright, A.H.: The Simple Genetic Algorithm and the Walsh Transform – Part I Theory, Part II The Inverse. Evolutionary Computation 6(3), 253–273, 275–289 (1998)
6. Vose, M.D.: The Simple Genetic Algorithm – Foundations and Theory. MIT Press, Cambridge (1999)
7. Vose, M.D.: Random Heuristic Search. Theoretical Computer Science 229(1-2), 103–142 (1999)

Imitation Learning in Uncertain Environments

Steffen Priesterjahn and Markus Eberling

Department of Computer Science, University of Paderborn, 33098 Paderborn, Germany
{spriesterjahn,markus.eberling}@upb.de

Abstract. This paper describes a new evolutionary learning method to handle the fast adaptation of a group of agents in an uncertain environment. The method is a result of our research towards the generation of intelligent agents in computer games and is inspired by the idea of social learning or cultural evolution. Thus, the agents try to adapt by the exchange of information about advantageous behaviours within the population. This paper evaluates the new approach by addressing the generation of competitive artificial players in a real-time action game.

1 Introduction

The requirement to learn online and to adapt in real time presents a challenging problem to artificial intelligence research. In our research towards artificial intelligence for computer games [1] we often face the challenge that our game agents have to adapt quickly in a very dynamic and uncertain environment that often contains unpredictable human players. This makes it very hard to quickly find out which actions or behaviours generally lead to an improved performance. In fact, when we tried to apply Q-learning [1], a typical online learning method, the results were not satisfactory. Because of the high volatility of the environment the experiences that were gained from the rewards of the learning agent were not solid enough to form the basis of the learning process. Therefore, we proposed the usage of a population of agents and their combined experiences as the basis of learning because it can greatly enhance the quality of the results and the stability of the learning process in the highly uncertain game environment[2].

In our previous work [1,2,3] we successfully used an evolutionary algorithm to breed well performing game agents that were initialised from player recordings. However, because of the exploratory nature of its variation operators, this approach tends to also generate defective and malfunctioning agents, i.e. agents that show random movements or other non-fluid behaviours. This might be no problem for scientific experiments, but in an ongoing game all game agents should show valid behaviours to not diminish the gaming experience. In addition, the method reacted quite sensitively to parameter changes. Therefore, this paper presents an improved learning method that combines the advantages of population-based learning with the ability to learn online by incorporating ideas from reinforcement learning. It is inspired by the concepts of social learning[4] and memetics [5] and follows the idea that in a group of individuals the low performing ones try to improve themselves by imitating the well performing ones. Therefore, we chose to call it *imitation learning*. We have already published some raw ideas of imitation learning in 2007 [3]. However, this paper presents the method in a much improved and more sophisticated form. Concerning related approaches, imitation learning bears several resemblances to the dynamic scripting method by Spronck et al. [7] in that it also determines the effectiveness of behaviour rules that are then shared between all agents. However, imitation learning is able to generate new rules by using it's mutation operator. Therefore, it can be used for learning from scratch. In addition, imitation learning sees each agent as a learning entity that imitates the best agents, whereas dynamic scripting uses a central rule base from which the agents are newly constructed after each adaptation step.

2 Imitation Learning

Imitation learning can in general be applied to all reinforcement learning problems or problems in which an agent has to learn to become competitive in an uncertain environment. In the proposed form it requires that the encoding of the behaviour of an agent is done as a set of rules - very similar to learning classifier systems. However, it is important to notice that - unlike in a learning classifier system - in our implementation there is only one rule that is executed in each time frame. The rule that is going to be applied is chosen as the one with the highest similarity to the currently encountered situation. If several rules have the same similarity, the one with the best fitness will be chosen. As it is usual in reinforcement learning we assume that the Markov property holds, so that the history of former states can be ignored. The adaptation process can be divided into four steps: evaluation, elite identification, rule replacement and individual adaptation / mutation. Algorithm 1 presents an overview of the approach.

Algorithm 1. Imitation Learning

```
inputs: $\mu, \sigma, \nu \in \mathbb{N}$, $\mu \leq n$
initialise $\nu$ agents
loop
    evaluate agents
    elite identification: determine the $\mu$ elite agents
    for all non-elite agents do
        choose a random role model from the elite agents
        select $\sigma$ rules from the role model
        replace rules
        mutate $\sigma$ worst rules
    end for
end loop
```

Evaluation: After initialisation the agents and their rule sets are evaluated in parallel for a certain timespan, whereas the performance of an agent is determined by the accumulated rewards over the evaluation phase and the rule values are initially determined by summing up the received rewards upon their application. However, the value of a rule is not independent from the other rules. The interplay between certain rules is usually very important for the behaviour of the agent. Therefore, we have to take the rules into account that have led to an advantageous situation in which the agent made a successful move. To do this we have adapted the policy evaluation algorithm that is known from the reinforcement learning field. For a set of rules R, let $v_0(r) \in \mathbb{R}$ be the initially sensed value of rule $r \in R$ after the evaluation phase. The *value* $v(r) \in \mathbb{R}$ of a rule $r \in R$ in a rule list R is then defined by

$$v(r) = v_0(r) + \gamma \sum_{r' \in R} p_{rr'} v(r'), \tag{1}$$

where $p_{rr'}$ is the transition probability between r and r' and $\gamma \in [0,1[$ is a discount rate. Let r^t be the rule that is chosen at some time frame t. Then, the transition probabilities $p_{rr'}$ are defined as

$$p_{rr'} = P(r^{t+1} = r' | r^t = r). \tag{2}$$

Therefore, $p_{rr'}$ is the probability that at the next time frame r' is chosen under the condition that r has been executed in this time frame. To gain these probabilities the transitions between the rules are counted during the evaluation timespan and then normalised. Equation 1 describes a system of linear equations that has the size of the underlying rule set. The solution of this system can be easily found or approximated by the appropriate algorithms, e.g. Gaussian elimination, fixed point iteration, etc.

Elite Identification: After the evaluation, the $\mu \in \mathbb{N}$ best performing agents, as indicated by their accumulated rewards, are identified as the *elite agents*. The others become the *imitators*.

Rule Replacement: The rule replacement is the crucial part of the adaptation mechanism. As the single rules are not independent from each other, many behaviours are encoded by a sequence of rule applications. A simple replacement of rules with low values by elite rules with high values will often lead to defective agents. Therefore, we devise a more careful mechanism that is inspired by results from memetics and viral marketing research about the acceptance and spreading of new ideas[8].

First of all, each imitator randomly selects just one elite agent as its role model, from which it receives new rules. Therefore, the incoming rule package is known to have successfully worked together. We propose to simply transmit the σ highest-valued rules of the chosen elite agent.

To maintain the coherence of the rule sets a new rule can only replace the rule that is most similar to itself. So, for each incoming rule, the imitator identifies the rule that it would apply in the most similar situation. These two rules then compete by

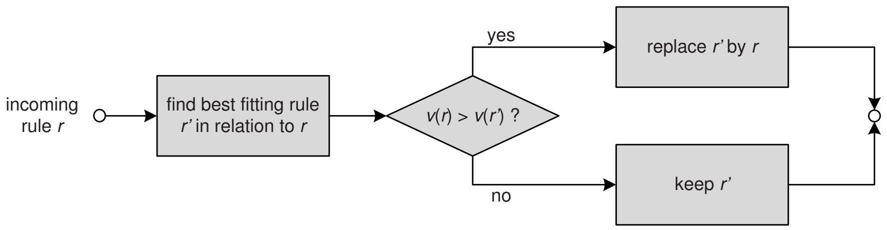


Fig. 1. Rule Replacement

their value. The new rule will only replace the old one, if it proposes a higher utility. Figure 1 illustrates this procedure.

Individual Adaptation / Mutation: For individual adaptation we simply propose to devise a mutation mechanism, though other adaptation techniques are possible. With respect to the uncertainty and high dynamics of our considered environments, the elite agents do not change their rules. This keeps the learning process more stable. They therefore can be seen as taking the role of the parents in an evolutionary algorithm.

3 Application to QUAKE III

This section describes how we successfully applied imitation learning to learn combat in the game QUAKE III (©1999, id software) - a very popular three-dimensional action game. As this approach is based on our previous evolutionary methods[1,2,3], the basic modelling of the used states, actions and rules resembles this work in most respects. The agents use regular grids for their state representation and use rule lists to encode their behaviour. A grid describes the current game situation as a mapping of the current relative vicinity of the observed agent to square areas. These areas can have three different values - empty, filled and opponent - that are based on their content. The agents are not able to look behind solid objects. Figure 2 presents an example for the construction of such a grid.

As required by the imitation learning method, the behaviour of an agent is encoded in a rule list, which contains rules that map grids to basic movement actions. Hence, each rule contains a grid that represents the state in which it should be applied and an action that proposes the next move, e.g. move left with speed v_x and forward with speed v_y, turn right by α degrees and attack. According to the current situation the best fitting rule of the rule list is determined by computing the Euclidean distance between the currently sensed grid and the grids that are proposed by the rules. For a better determination of the similarity between the grids, all grids are smoothed by a Gaussian filter - a convolution with a 5×5 Gaussian filter matrix - before they are compared. The agents work at a rate of ten times per second.

Initialisation: The agents are initialised with behaviour rules from a recorded player to start at a better position in the search space. These rules simply state what movement

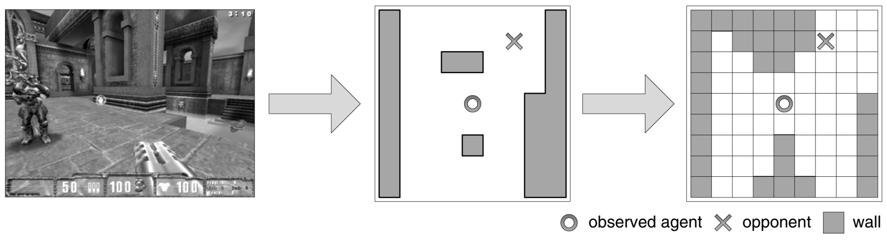


Fig. 2. Grid Computation

the player made in a corresponding game situation. In our experiments we used the built-in QUAKE III agent as the source for the recording because it also serves as the benchmark for the performance of the agents.

Evaluation: All agents play in parallel in different game sessions and are evaluated for one minute of combat gameplay against the built-in QUAKE III agent on a small training map. The QUAKE III agents are artificial players that are supplied by the game and which use some randomised hard-coded strategies and therefore present constant opponents and are thus usable as a benchmark. The rewards and the initial rule values are determined by the damage that the agent applied to its opponents minus the damage that it received when the respective rule was applied. As stated above, the overall *performance* of an agent in combat is measured by computing the amount of damage that it inflicted on the other game characters minus the damage that it received in the whole evaluation timespan.

Individual Adaptation / Mutation: The imitators mutate the σ rules with the lowest utility. To reduce the number of parameters, we use the same value σ as for the number of transmitted rules because the purposes of the parameters are somewhat related. We assume, that the recording that was used for initialisation already contains all important states and, therefore, apply no mutation to the grids and only add small changes to the proposed commands, e.g. randomly move into another direction or add a small Gaussian distributed value to the proposed turn angle. In our implementation, the agents will only mutate a rule, if it has a negative utility. A negative utility indicates that the rule has led to a situation in which the agent has received damage. Therefore, it is reasonable to do something else in the respective situation.

4 Experimental Setup

The given approach has several degrees of freedom. Fortunately, we could use previous results to assign good values to most parameters. Tables 2a and 2b give an overview of the parameters and their respective values. We stopped all experiments after 80 minutes because we expected our online adaptation algorithm to be able to adapt within a short time span. Parameters 8-12 could not be directly assigned with good values.

The *discount rate* γ specifies how much the value of a rule should depend on the rules which were executed afterwards. It should not be too high or too low for obvious

Table 1. Parameter Setup

#	parameter	value
1	grid size	15
2	grid field size	100
3	rule list size	100
4	mutation probably	0.1
5	evaluation timespan	60s
6	runs per experiment	20
7	experiment length	80 min

(a) Assigned Parameters

#	parameter	value
8	population size	ν
9	elite size	μ
10	number of transmitted rules	σ
11	discount rate	γ
12	mutation rate	π

(b) Open Parameters

Table 2. Experimental Setup (changed parameters are bold)

experiment	population size ν	elite size μ	sent rules σ	discount rate γ	mutation rate π
base experiment	32	4	40	0.7	0.1
1.1	**8**	4	40	0.7	0.1
1.2	**16**	4	40	0.7	0.1
1.3	**64**	4	40	0.7	0.1
1.4	**128**	4	40	0.7	0.1
2.1	32	**1**	40	0.7	0.1
2.2	32	**8**	40	0.7	0.1
2.3	32	**16**	40	0.7	0.1
3.1	32	4	**5**	0.7	0.1
3.2	32	4	**20**	0.7	0.1
3.3	32	4	**60**	0.7	0.1
3.4	32	4	**80**	0.7	0.1
4.1	32	4	40	**0.0**	0.1
4.2	32	4	40	**0.4**	0.1
4.3	32	4	40	**0.9**	0.1
5.1	32	4	40	0.7	**0.0**
5.2	32	4	40	0.7	**0.01**
5.3	32	4	40	0.7	**0.5**

reasons. The *number of transmitted rules* σ specifies how many rules the elite agents send to the other agents in the population after an evaluation phase. If it is too high, the population might become more uniform in the course of the adaptation. If it is too low, only some of the most important rules might be transmitted and some crucial rule might be missing. The *elite size* specifies the number of chosen elite agents in each adaptation step. It should be big enough to handle the uncertainty of the environment but also small enough to generate an acceptable learning speed. The *population size* ν should be big enough to statistically handle the high dynamics of the game and to have a high enough diversity of rule lists - especially at the beginning. Furthermore, each agent has to adapt to its own opponent. So, some agent might experience some more valuable events, which helps the others. Finally, the *mutation rate* describes the degree of variation that is added by the mutation operator. In particular it determines the probability that a part of a rule is changed.

Table 2 shows the experiments that were conducted and their respective parameter setup. We chose these values as a result of a series of former experiments and tests. The setups were organised in a way in which first a base setup was chosen and then each parameter was systematically changed to detect its influence. Though some of the parameters are likely to not be independent from each other, this method is very helpful for obtaining an understanding of their general meaning. Each set of experiments represents such an examination of one parameter.

Some setups were using some extreme values. Of particular interest are the setups 4.1 and 5.1. In setup 4.1 the discount rate γ was set to zero to see how the algorithm performs if the discounted evaluation of the rules is switched off and the value of a rule is just the immediate gained reward upon its execution. Setup 5.1 switches off the mutation of the worst rules to detect how much of the performance gain is created by the assembling of good, fitting rules and how much is gained by changing bad performing rules.

5 Results

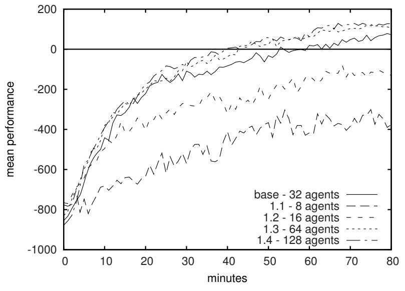
The results of the conducted experiments were very successful as the imitation learning approach was able to generate the same quality of solutions as our previous evolutionary approach but showed a much higher robustness and a much lower tendency to generate defective agents. The best experiments reached a mean performance of around zero after thirty to forty minutes. This means that the mean of all agents performs as good as the opponents and that about 50% of the agents are winning their matches.

In contrast to offline learning, where the result is usually an agent which is selected as the best generated agent after the learning process, online learning should produce a whole population of competitive agents. Therefore, the following analysis concentrates on the mean performance of the agents in each adaptation step, averaged over 20 runs.

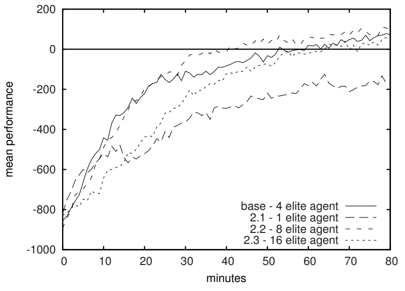
Figure 3a shows the influence of the population size on the algorithm. It shows that a sufficient amount of agents is needed to make the algorithm work. Apparently, with the used settings, 8 agents are not enough to obtain competitive behaviour. Using 16 agents increases the mean performance, but the agents still do not reach a mean performance of around zero. Not until a population size of 32 agents is used, the algorithm works well. Interestingly, the usage of even more agents does not significantly increase the mean performance. However, a higher population size gives the approach more statistical stability. It is not disadvantageous to use as many agents as possible. One reason for the poor performance when using 8 or 16 agents could be that the pool of rules the agents start with is simply not diverse enough to obtain competitive behaviour.

The experiments in set 2 varied the elite size μ. In comparison to plain evolutionary algorithms, μ corresponds roughly to the number of selected parents because it determines the number of elite agents from which the others incorporate new knowledge. Therefore, the effect of changing μ is similar to changing the selection pressure or the degree of exploitation of the approach. The results in figure 3b show that μ should be chosen greater than one. The experiment with $\mu = 1$ performs significantly worse than the other experiments in this set. Though always copying the single best agent seems to be quite reasonable, the drawback of this approach is that the game and thus the environment of the agents is too uncertain and dynamic to specialise so much. Therefore, μ has to be chosen according to the uncertainty and dynamics of the given environment. In a completely deterministic world, an elite size of $\mu = 1$ should produce the fastest convergence, but might only lead to a local optimum. The other experiments eventually reached about the same performance. However, because of its lesser degree of exploitation, the experiment using $\mu = 16$ lagged behind the ones using an elite size of 4 or 8. Therefore, μ should be chosen not too low but also not too high.

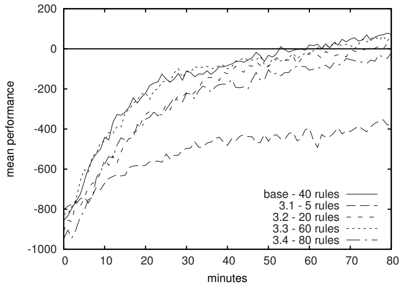
As we already mentioned, the algorithm has shown a high robustness against parameter changes. Especially, the number of transmitted rules σ - as seen in figure 3c - has only a small influence, when set to sane values. Only if the number is too low - as in setup 3.1 - the algorithm did not perform well. All other setups reached about the same performance as the base setup. σ is used at two points of the adaptation algorithm. The first point is the number of rules that are selected for imitation. Our approach always selects the σ best rules. The imitators incorporate these rules by the rule replacement method that we specified above. This method compares the values of the incoming rules and compares them to the rule which should be replaced. If σ is very high, the additionally



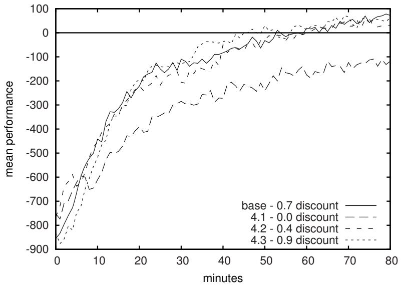
(a) Set 1: Variation of ν



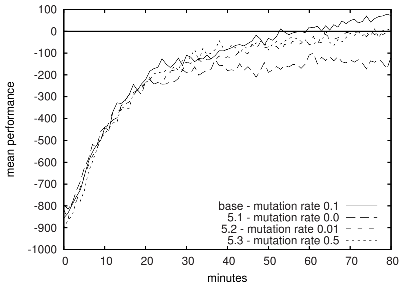
(b) Set 2: Variation of μ



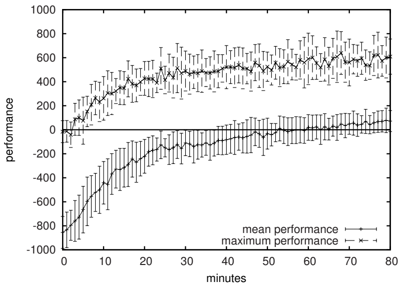
(c) Set 3: Variation of σ



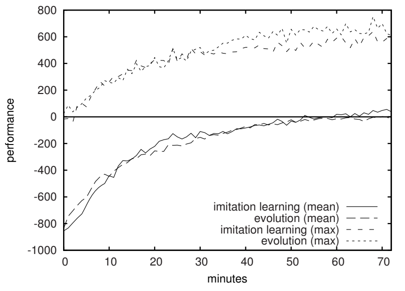
(d) Set 4: Variation of γ



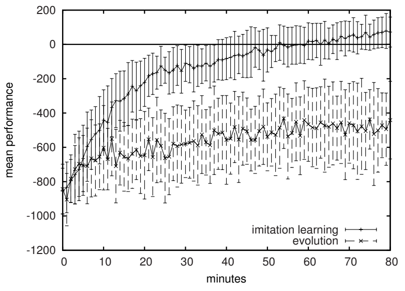
(e) Set 5: Variation of π



(f) Results of the Base Setup



(g) Comparison to Evolution



(h) Comparison (Both with 32 Agents)

Fig. 3. Results

selected rules tend to have very low or even negative values which makes them less and less likely to replace an old rule. Furthermore, if the low valued rules replace some rule, they will always replace a rule that by itself already had a low value. The second point where σ is used, is the number of selected rules for mutation. Here, our approach always selects the σ worst rules. However, the mutation operator will be only applied, if the value of the respective rule is below zero. Therefore, setting σ to a very high level, will not damage the well performing rules.

In set 4 (see figure 3d) of the experiments we varied the discount rate γ to detect its influence on the algorithm. Again, imitation learning shows a high robustness against parameter changes. All experiments with a discount rate of $\gamma > 0$ produced competitive agents. To show that the discounting and thus the reinforcement learning-based part of the algorithm has some influence at all, we also made experiments that used a discount rate of 0.0. This effectively turns off the policy evaluation. In this case figure 3d shows that the agents perform significantly worse. However, the algorithm still improves the agents based upon the sole rule execution rewards and the fitness of the agents. It just cannot reach the last bit of performance increase that is gained by relating the rule values against each other because it might ignore momentarily disadvantageous rules that might lead to advantageous situations. Another reason why the algorithm still performs quite well is that $\sigma = 40$ of 100 rules are transmitted. This increases the probability that important rules that do not have a direct impact are still transmitted.

Finally, the experiments of set 5 examined the influence of the mutation rate π. The results show that the variation of the mutation rate has only a very small effect on the obtained performance. This has several reasons. First, the mutation is only important as long as rules with negative values exist, as they are the only ones which will be mutated. As a consequence of the discounted evaluation of the rules, it will be less and less likely that rules with negative values exist, when the agent begins to defeat its opponent. Second, the rules stem from a recording of a player and therefore need only slight adjustments to work well. In addition, as we only mutate the commands and not the grids, the rules always stay somewhat sane. Only few information can be destroyed. In experiment 5.1 the mutation rate was set to zero, which effectively turns off the mutation of bad rules. This experiment produced a significantly lower performance. Therefore, the mutation is important. In fact, this experiment shows how much performance can be obtained by just finding the best fitting collection of recorded rules. The remaining gap is closed by small adjustments to optimise the rules themselves.

For better statistical judgement figure 3f provides the standard deviation of the base setup for the mean and the maximum performance of the agents. The figure also shows that even when adding the standard deviation most experiments end at about the desired zero mean performance after at maximum one hour.

Finally, figure 3g compares the mean and the maximum performance per generation of the base setup for imitation learning to the best setup that we could get with our previous evolutionary approach. There, imitation learning reaches about the same performance, which confirms that, though imitation learning is much more focused on exploitation, it is still able to produce the same quality of solutions. However, in contrast to the population size of 32 that was used by our base setup for imitation learning, the evolutionary algorithm needed 60 agents to reach this result. So, figure 3h shows the mean performance of both approaches when just 32 agents are used. Here, imitation learning significantly outperforms the exploratory evolutionary technique. As a general result, imitation learning has proven to reliably deliver very good results and to be very robust in terms of parameter dependency and environment uncertainty.

Concerning the defectiveness of the agents, the pure plots are not enough to give a real indication because they are composed of such a high number of experiments. Though the higher derivation of the results of the evolutionary method gives an indication of its inferior results, we had to rely on our observations during the training process to judge the quality of the solutions that were gained by imitation learning. There, the number of defective agents that appeared in the course of the adaptation process with the imitation learning approach decreased very fast. Already after the first three to four minutes no defective agents could be seen, though some low performing agents still showed up from time to time. When observing the gaming behaviour of the produced agents, all successful experiments showed more or less the same progression[1]. Right after initialisation the behaviour was a bit awkward and clumsy. Then, with each adaptation pass the behaviour of the agents became sharper and more refined. After about five minutes the first agents could defeat their opponents. As in the evolutionary approach, in this phase the agents were almost mirroring the behaviours from the recording that was used for their initialisation. Later, as the game progressed, the agents started to take more freedom in their movements and showed more sophisticated behaviours. For more details we refer to Steffen Priesterjahn's PhD thesis on this topic [1].

6 Conclusion

The presented results show that it is possible to create a population-based online adaptation method that is based on the idea of social learning. Imitation learning is able to quickly generate competitive game agents without producing defective agents. The method is especially well-equipped for improving the mean performance of a population of agents. The most important result is the very high robustness of imitation learning which is based on the usage of the experiences of a population of agents and the adaptive variation operators for rule replacement and mutation. This makes this approach very applicable in practical problems, such as the automatic adaptation in computer games.

References

1. Priesterjahn, S.: Online Adaptation and Imitation in Modern Computer Games. PhD thesis, University of Paderborn (2008)
2. Priesterjahn, S., Kramer, O., Weimer, A., Goebels, A.: Evolution of Human-Competitive Agents in Modern Computer Games. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2006), pp. 777–784. IEEE Press, Los Alamitos (2006)
3. Priesterjahn, S., Weimer, A.: An Evolutionary Online Adaptation Method for Modern Computer Games. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007) (2007)

[1] See `https://chaos.cs.upb.de/il.wmv` for a demonstration.

4. Conte, R., Paolucci, M.: Intelligent Social Learning. Journal of Artificial Societies and Social Simulation 4(1), U61– U82 (2001)
5. Blackmore, S.J.: The Meme Machine. Oxford University Press, Oxford (2000)
6. Miles, C., Louis, S.J., Cole, N., McDonnell, J.: Learning to Play Like a Human: Case Injected Genetic Algorithms Applied to Strategic Computer Game Playing. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2004) (2004)
7. Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., Postma, E.: Adaptive Game AI with Dynamic Scripting. Machine Learning 63(3), 217–248 (2006)
8. Chielens, K., Heylighen, F.: Operationalization of Meme Selection Criteria: Methodologies to Empirically Test Memetic Predictions. In: Proceedings of the Joint Symposium on Socially Inspired Computing (AISB 2005), pp. 14–20 (2005)

Using Ants' Task Division for Better Game Engines – A Contribution to Game Accessibility for Impaired Players

Alexis Sepchat, Romain Clair, Nicolas Monmarché, and Mohamed Slimane

Laboratoire d'Informatique de l'Université François Rabelais de Tours, France
Équipe Handicap et Nouvelles Technologies (HaNT)
{alexis.sepchat,romain.clair,nicolas.monmarche,
mohamed.slimane}@univ-tours.fr

Abstract. Designing a relevant artificial intelligence engine for video games does not always consist in finding the best solution (best opponent, best path, etc.): it can sometimes consist in offering the player the best gaming experience. Such a good experience is linked with the difficulty level of the proposed challenge. A game that is too easy will be boring whereas a game that is too difficult will be stressful. So, to be interesting for everyone, an artificial intelligence engine should provide an adaptive game level for every player. This game tuning is particularly prominent in the especial case of accessible games for impaired player. In this paper we show that the task division model based on ant colonies can be an interesting way to provide adaptive behaviors in game engines for simple one-player games.

1 Introduction

Despite their limited cognitive abilities, real ants are social insects that are able to solve complex tasks (ant-hill building, food search, etc.) principally thanks to their social interactions which is the synergy (ant colony ability is greater than the sum of individual ants' abilities). Several mathematical models have been proposed in order to simulate social behaviors of ants [1] and are used in computer science in order to conceive solving methods for complex problems such as clustering [2] or combinatorial optimization [3].

Their robustness and their effectiveness are characteristics of these models. Indeed, as real ant colonies are able to adapt themselves to their environmental changes, ant colony based algorithms are also able to perform adaptations according to initial or continuous data. This particular aspect has led us to consider an artificial ant colony for a self adaptation engine in computer games. Indeed, a player can be considered as a disruptive agent in a given environment and the ant colony has to react in order to prevent him/her from winning. Moreover, the higher a stimulus intensity is, the more efficient the ant colony is (we will explain why in section 3). So, the stronger the player is, the more efficient the ant colony will be. Finally, thanks to artificial ants, game level will depend on the player because the game level will adapt itself.

Ant colonies have already been used in games, not for level adaptation, but for Non-Player Characters' moves [4] (pathfinding or band move). In this article we will focus on the task allocation model and its introduction in an accessible game AI engine able to adapt the difficulty level to the player.

This article is organized as follows: in section 2, we will introduce computer games accessibility. Section 3 focuses on the problem of allocating tasks and the associated mathematical model. The game developed from this model is explained in section 4. Finally, section 5 will conclude this paper and proposes further works.

2 Computer Games Accessibility

2.1 What Is Computer Games Accessibility?

Currently, computer games are a full-fledged element of our communication and information society. That is why everyone, whatever his or her physical or cognitive abilities, should be able to play them. Indeed, for several years, computer games have become one of the main types of numerical entertainment. According to a recent French survey[1], nearly half of all French homes play computer games. Even if entertainment is fundamental, it is not the only aspect. Computer games are wonderful pedagogical and social integration tools [5,6]. That is why computer games accessibility is really an ethical, legal, financial and technical challenge [7]. Defined for the first time in 2004 by the Game Accessibility Special Interest Group of the International Game Developers Association (GA-SIG IGDA) as:

> "the ability to play a game even when functioning under limiting conditions. Limiting conditions can be functional limitations, or disabilities - such as blindness, deafness, or mobility limitations."

Game accessibility can concern everyone. The definition takes into consideration "limiting conditions" which deals with, but it is not limited to, impaired players [8].

Currently, all mainstream games can be considered as inaccessible (even if a few of them are partially accessible: accessible for a particular impairment or only a part of the game is accessible). In a general way, when an impaired player tries to play a computer game, he or she meets two kinds of problems: interaction problems and level problems.

2.2 Interaction Problems

Interaction problems deal with the problems encountered by a player in its interaction with the game: problems in information acquirement from the game or

[1] The French market of computer games:
`http://www.afjv.com/press0611/061122_marche_jeux_video_france.htm`

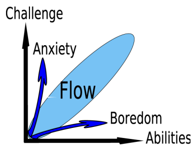


Fig. 1. Notion of "flow" in games by Chen Jenova

problems in transmitting commands to the game. A way to solve these problems is to develop multi modal applications. Multi modality consists in proposing several game representations based on different modalities (graphic, audio: speech synthesis, haptic: force-feedback, tactile: braille terminal) and several game controllers also based on different modalities (joypad, joystick, speech recognition, breath etc.). So, according to his or her abilities, the player can choose which modality to use.

2.3 Level Problems

Level problems deal with problems of complexity but also with problems of game speed, game comprehension (learning each character's role, learning the use of controllers, etc.), etc. So level problems are larger than cognitive impairment. They concern every player whether they are impaired or not. So level problems are much more difficult to take into account than interaction problems because, even if the level can prevent the user from playing the game, it is also what makes the game interesting.

In order to illustrate the role of level in games, Chen Jenova [9] uses the notion of "flow" defined in 1975 by the psychologist Csikszentmihalyi as "the feeling of complete and energized focus in an activity, with a high level of enjoyment and fulfillment"[10]. So, this feeling represents the state in which the player takes pleasure in playing. According to Jenova, it depends on the abilities of the player and the challenge of the game (Cf. figure 1) [11]. So the game must not be too difficult nor too simple according to the player's abilities in order to be interesting.

This representation points out the difficulty to create UA-Games[2] that are interesting for both players with and without impairment. Thus the game level has to adapt dynamically according to player abilities.

Such a mechanism is called "adaptive game AI". According to Pieter Spronck in [12], "adaptive game AI" can be defined as game AI with the ability of self correction (*i.e.* the ability to resolve faulty agent behavior) and with the ability of creativity (*i.e.* the ability to adapt successfully to changing circumstances). Different techniques can lead to adaptive AI:

offline learning: what the game learns while it is not being played. This can be learning from samples or learning by self-play.

[2] Universally Accessible games (`http://www.ics.forth.gr/hci/ua-games/index.html`)

supervised learning: what the game learns while it is being played by a human. It implements changes to the game AI by processing immediate feedback on any decision that the game AI makes.
online learning: what the game learns while it is being played by a human in order to adapt the game to the player's tactic, style, skill, etc.

In order to solve level problems in the case of accessible games, we will focus on online learning methods. Such methods need the following four computational requirements : speed, effectiveness, efficiency and robustness [12]. Nevertheless, these four requirements are part of the main interest of ant-based algorithms. It is why, from the observations of auto adaptation skills in ant colonies and a mathematical model of such behavior, we have decided to use this characteristic to create online learning and dynamic adaptation within a simple game.

3 Division of Labor, Task Allocation Problem and Bonabeau's Associated Model

3.1 From Observations ...

An ant colony is able to perform efficiently and simultaneously a large number of various tasks thanks to an efficient and robust sharing out mechanism (search for food, ant-hill building, *larvae* breeding, waste sorting, etc.). Each task in this system has to be performed to ensure colony survival. Despite their simplicity, each ant seems to choose the best task to perform (allocation notion) among a list of possible tasks (division of labor notion) according to the colony's needs. This choice can not be made randomly. Collective intelligence has to influence it by giving information about the priority of each task.

3.2 ... to a Division of Labor and Allocation Tasks Model ...

From these informations, Bonabeau *et al.* [13] have proposed a mathematical model of these behaviors. Their model relies on reactive agents: each ant reacts according to information extracted from its direct environment. So, it has only a local knowledge and can not act in anticipation. A representation of ants' behavior based on response thresholds and *stimulus* intensities has been defined to simulate ants tasks choices. This representation using probabilities introduces the following notions:

***stimulus* intensity S_i:** in order to ensure the continuous existence of the nest, a set of tasks has to be performed by the colony. These tasks are associated with a *stimulus* and a positive or null value, called *stimulus* intensity, representing task priority. The higher the value, the more important the task. We denote S_i with the *stimulus* intensity associated with the task i;
response threshold $\theta_{i,j}$: each ant in the colony has a set of response thresholds associated with each *stimulus* (thus each task). A response threshold represents a *stimulus* intensity level from which the corresponding task can

be chosen by the ant. Hence, the lower a *stimulus*-associated response threshold is, the greater is the chance to perform the corresponding task. We denote $\theta_{i,j}$ with the response threshold for the task i (thus the *stimulus* intensity S_i) and the ant j;

probability $T_{\theta_{i,j}}$ to perform a task i: we denote $T_{\theta_{i,j}}$ with the probability that an ant j starts a task i. This probability depends on the ant j response threshold associated to the task i: $\theta_{i,j}$ and *stimulus* intensity associated with i task: S_i. Generally, if the *stimulus* intensity is negligible compared to the response threshold, the probability to perform the task should be near 0. Contrarily, it should be near 1 if the response threshold is negligible compared to the *stimulus* intensity. In our engine we use the following function (Cf. equation 1):

$$T_{\theta_{i,j}} = \frac{S_i^2}{S_i^2 + \theta_{i,j}^2} \tag{1}$$

probability p to stop the current task performance : when an ant is performing a task, it has the possibility to stop at each time step. This possibility is given by the probability p shared by all the colony. So, on average, an ant will perform a task during $1/p$ time step.

ant colony efficiency α: ant colony efficiency is a global parameter shared by all ants of the colony for all tasks.

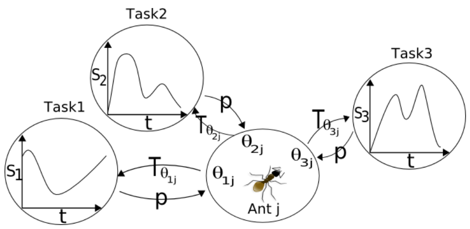


Fig. 2. Automaton-like model's behavior

With just a few parameters, a simple task allocation model can be set. Its dynamic aspect relies on *stimulus* intensity variations. This variation takes two parameters into consideration:

natural increase δ_i: in a general way, *stimulus* intensity increases over time if no ant performs the associated task. So each time step, the intensity will increase by δ_i. In the case of an AI game engine, there is no natural increase. It is replaced by player's actions.

performed task: the only way to decrease *stimulus* intensity is to perform the corresponding task. The diminution depends on the number of ants performing the task during the period and their effectiveness. We denote $N_{act,i}$ with the number of ants performing the task i during the period.

$$S_i(t+1) = S_i(t) + \delta_i - \frac{\alpha.N_{act,i}}{N} \tag{2}$$

Moreover, it is necessary to scale the amount of work performed by the $N_{act,i}$ ants by the total number of ants N in the colony in order to translate the fact that the larger the colony is, the greater the amount of work to perform will be.

Nevertheless, model performances vary a little bit from real ants' behavior. For instance, in Nature, ants are able to specialize in a particular task. This specialization can lead to morphological or behavioral alterations. But in all cases, if an ant is a task j specialist, it will be more efficient at performing it and it will have a preference to perform it. This is why Theraulaz *et al.* have extended the previous mathematical model to integrate specialization mechanisms.

specialization: in Bonabeau's mathematical model, efficiency is a global parameter shared by all the ants of the colony. So, specialization can not modify ant efficiency to perform a particular task. The only way to integrate such a mechanism is to take into consideration ant predilection for a particular task. Theraulaz *et al.* have introduced a learning coefficient ξ and a forgetting one, φ, which allow to modify ants' response thresholds.

$$\theta_{i,j}(t+\Delta t) = \theta_{i,j}(t) - \xi\Delta t \tag{3}$$
$$\theta_{i,j}(t+\Delta t) = \theta_{i,j}(t) + \varphi\Delta t \tag{4}$$

3.3 ... and Its Implementation

Some elements are missing in Bonabeau's model description and we have to perform further studies to implement such a model. Among these missing elements, we will focus on the task selection strategy and the move strategy. These two kinds of strategies lead to important variations of system performances in terms of efficiency, reactivity, etc.

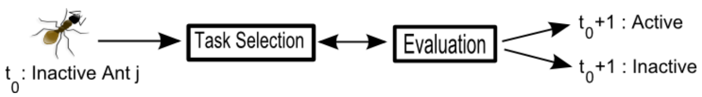


Fig. 3. Task selection and move strategies illustration. Selection consists of choosing one of the available tasks. Evaluation consists of determining if the ant has to perform the selected task or not during the next time step. This choice is based on the stimulus intensity and the corresponding ant's response threshold. Task selection strategy influences the Selection step whereas move strategy influences the Evaluation step.

Tasks Selection Strategy. At each time step, an inactive ant has to determine if it will remain inactive during the next time step or if it will perform a task. This choice is based on the mechanism given in figure 4 where the inactive ant selects one task and evaluates it. Several selection strategies can be used. For

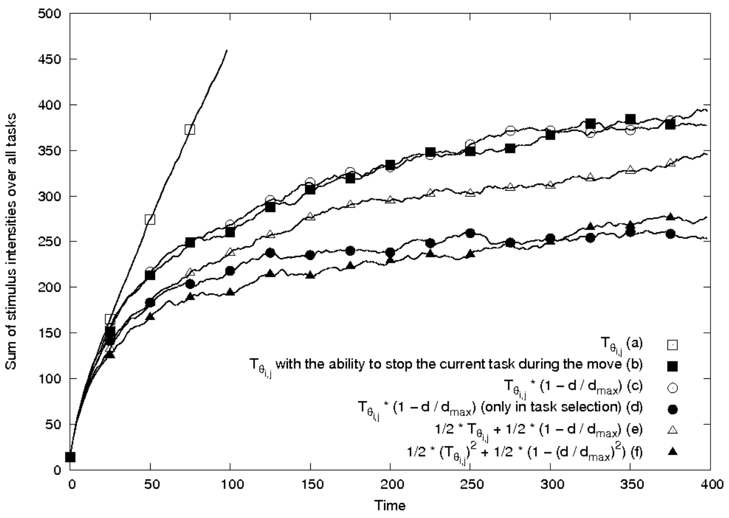


Fig. 4. Model behaviors according to several move strategies with the same predetermined player's actions (averaged over seven runs) functions a and b use the initial Bonabeau's action probability ; functions c and d balanced Bonabeau's action probability with distances; functions e and f consider the sum of Bonabeau's action probability and distances

example, an ant j can evaluate only one task i and according to its probability to perform the task $T_{\theta_{i,j}}$, and decide to perform it or not (cf. case a figure 3). Contrarily, it can evaluate all the tasks until one of them is chosen or none are accepted (cf. case a figure 4). These two strategies lead to an important variation of inactive ants rate thus an important variation of the whole system's efficiency. In the first case, the global rate of inactive ants is higher so the system's efficiency is lower.

In our case, the colony must be efficient enough to represent a good challenge for skilled players, but the inactive ants rate must vary enough to match the players' performance. A good compromise has been found : we evaluate only one task and we use cumulated probabilities. Thanks to this mechanism, a random number permits the selection of one task. This task is selected by taking into account the different priority levels. This moderates the inactive ants rate.

Move Strategy. During the game, the ant processes are visible because each artificial ant represents a Non-Player Character, so we need to implement realistic moves. Several strategies can be considered (cf. figure 4). For example, should distance between a task and an ant's position have an influence on ants' choices besides *stimulus* intensities and response thresholds? Can an ant decide to change its current task during its move? etc.

To answer these questions, we have computed the sum of *stimulus* intensities for each time step. A high value means that the colony is overwhelmed. Among these strategies, we have studied results obtained with the initial Bonabeau *et al.*

model, which does not take into account distance (with or without the ability to stop a task during its move - squares on the figure 4), and models taking distance into account (discs and triangles on figure 4).

Obviously, distance has to influence ants' choices. Indeed, in a 2D environment, the ant colony efficiency and reactivity are increased when distance is considered. We select several of these strategies to manage difficulty level (player keeps the possibility to choose the initial level : easy (b), medium (d), hard (f)).

4 The Game

4.1 Game Accessibility

Using this ant's task allocation model, we have created a game that is accessible for most of players. Firstly, game accessibility is ensured thanks to multi modality. Indeed, we have developed several game representations based on different modalities such as graphic, tactile (braille display) and audio. Several game controllers such as keyboards and braille displays are also available. Secondly, our ant colony based algorithm solves the level problems.

4.2 Tasks Allocation Game

The task allocation game we have developed is based on buffers. Indeed, the player tries to fill buffers whereas the ant colony tries to empty them. Based on

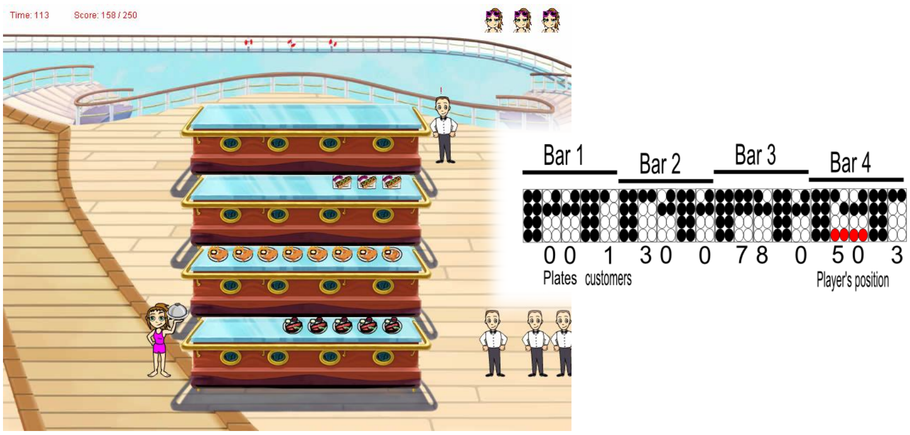


Fig. 5. Game screenshot and an associated Braille representation (on a 20 braille cells display). Each task's information is translated onto five braille cell groups : a full one for the beginning of the group - two for the number of plates (each plate is associated with a value between 1 and 10 respectively if it is almost empty or full and a bar can contain up to 8 plates so the value varies from zero to 80) - a full one - and one for the number of customers. Braille figure representations use the sixth upper splinters of each cell (3x2 matrix). The fourth splinters line shows the player's position.

this principle, several gameplays can be envisioned. We focus on the following one : the action takes place in a fast food restaurant where a waiter (*i.e.* the player) has to serve different meals to several consumers (*i.e.* the ants of the colony). The aim of the game is to satisfy consumers (*Cf.* figure 5). The score is the maximum quantity of food found simultaneously on each bar. The player loses a point life if a consumer is not satisfied. This very simple gameplay allows everybody to understand the game (children, cognitive impaired players, etc.).

The player can move from one bar to the neighboring ones and he or she can add a plate to the current one. The braille representation of the game (Cf. figure 5) permits the player to know, for each bar, so for each task, the number of plates (thus task stimulus intensity) as well as the number of customers (or the number of ants performing the task) (0123456789 ⠚ ⠁ ⠃ ⠉ ⠙ ⠑ ⠋ ⠛ ⠓ ⠊).

5 Conclusion and Further Works

Even if real ants do not exhibit playing behaviors such as, for example, felines [14], observing them has led us to design an efficient and adaptive AI engine for games. This AI engine is based on a few, specific global parameters (six). It's able to adapt the game's difficulty level according to the players, so the game becomes well-adapted to every player whatever their age or their physical or cognitive abilities. Moreover, used in collaboration with multi-modal interfaces, it solves the two problems of game accessibility : interaction and level. Our experience with visual impairment leads us to principally conceive modalities focusing on this impairment : braille terminal and audio representations[15,16,17,18]. Nevertheless, new interfaces, and in particular new controllers can be imagined to make our game accessible for motor-impaired players.

The game developed on the task allocation model is a simple one. Nevertheless, the underlying principle is used in school with young children to teach how to count (count how many plates there are on each bar; to compare which bar has more plates; etc.) and many games, or more generally, many computer science problems, make reference to this mechanism. For example, RTS games where several resources have to be gathered and a player must delegate each task to the villagers. This model has been introduced to manage villagers' allocation in the Boswars game[3] and in the Stratagus engine that it uses.

Currently several tests have been led with unimpaired adult players. Results are encouraging and tests with children and visually impaired players are being prepared. Tests with cognitive impaired players are also envisioned.

An implementation of other tools dealing with online learning such as evolutionary algorithms, neural networks, reinforcement learning, dynamic scripting, etc. are also envisioned [12,19]. This will lead to a real comparison of our engine performance. Moreover this mathematical model based on response thresholds has found other applications. For example, drawing a parallel between the *stimulus* intensity and the similarity of several elements in the same neighborhood, it can be used in clustering [20].

[3] Boswars website : `http://www.boswars.org/`

Finally biomimetic algorithms are a good candidate to conceive artificial intelligence in accessible computer games. Linked to accessible devices progress and multi modality, it will permit the improvement of game accessibility in order to create UA-games and a sharable game universe for all players with or without any impairments.

References

1. Garnier, S., Gautrais, J., Theraulaz, G.: The biological principles of swarm intelligence. Swarm Intelligence 1(1), 3–31 (2007)
2. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1(2), 95–113 (2007)
3. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, USA (2004)
4. Dunn, J.: Ant colony organization for mmorpg and rts creature resource gathering. In: Rabin, S. (ed.) AI Game Programming Wisdom, vol. 3, pp. 495–505. Thomson Delmar Learning (2005)
5. Mitchell, A., Savill-Smith, C.: The use of computer and video games for learning: a review of the literature (2004)
6. Gee, J.P.: Why video games are good for learning? In: Keynote address at Curriculum Corporation 13th National Conference (2006)
7. Bierre, K., Chetwynd, J., Ellis, B., Hinn, M., Ludi, S., Westin, T.: Game not over: Accessibility issues in video games (2005)
8. Grammenos, D., Savidis, A., Stephanidis, C.: Ua-chess: A universally accessible board game. In: Proc. of the 3rd Int. Conf. on Universal Access in Human-Computer Interaction, Las Vegas (2005)
9. Jenova, C.: Flow in games. PhD thesis, University of Southern California (2001)
10. Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. Harper Perennial, London (1990)
11. Archambault, D., Ossmann, R., Gaudy, T., Miesenberger, K.: Computer games and visually impaired people. UPGRADE 8(2), 43–53 (2007)
12. Spronck, P.: Adaptive Games AI. PhD thesis, Maastricht University (2005)
13. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
14. Hölldobler, B., Wilson, E.: The Ants. Springer, Berlin (1990)
15. Sepchat, A., Monmarché, N., Archambault, D., Slimane, M.: Accessible video games for visually impaired children. In: The Third Annual Int. Conf. in Computer Game Design and Technology, Liverpool, UK, pp. 58–67 (2005)
16. Sepchat, A., Monmarché, N., Slimane, M., Archambault, D.: Semi automatic generator of tactile video games for visually impaired children. In: 10th Int. Conf. on Computers Helping People with Special Needs, Linz, Austria, pp. 372–379
17. Sepchat, A., Monmarché, N., Slimane, M.: Accessible card games for visually impaired players. In: Eizmendi, G., José Miguel Azkoitia, G.C. (eds.) Proc. AAATE (European Conf. for the Advancement of Assistive Technology in Europe), San Sebastian, Spain, vol. 20, pp. 802–806. IOS Press, Amsterdam (2007)
18. Puret, A., Archambault, D., Monmarché, N., Slimane, M.: A simple game generator for creating audio/tactile games. Technology and Disability (2006)
19. Spronck, P.: Dynamic scripting. In: Rabin, S. (ed.) AI Game Programming Wisdom, vol. 3, pp. 661–676. Thomson Delmar Learning (2005)
20. Schockaert, S., De Cock, C.C.M.: Efficient clustering with fuzzy ants (2004)

A Set-Based Particle Swarm Optimization Method

Christian B. Veenhuis

Fraunhofer IPK, Dept. Security Technology, Pascalstr. 8-9, 10587 Berlin, Germany
veenhuis@googlemail.com

Abstract. The representation used in Particle Swarm Optimization (PSO) is an n-dimensional vector. If you want to apply the PSO method, you have to encode your problem as fix-sized vector. But many problem domains have solutions of unknown sizes as for instance in data clustering where you often don't know the number of clusters in advance.

In this paper a set-based PSO is proposed which replaces the position and velocity vectors by position and velocity sets realizing this way a PSO with variable length representation. All operations of the PSO update equations are redefined in an appropriate manner. Additionally, an operator reducing set bloating effects is introduced.

The presented approach is applied to well-known data clustering problems and performs better as other algorithms on them.

1 Introduction

In recent years a swarm-based optimization methodology called Particle Swarm Optimization (PSO) has developed. Usually, the representation used in PSO is an n-dimensional vector. Furthermore, all vectors of the swarm have the same dimensionality. If you want to apply PSO you have to encode your problem as fix-sized vector.

The question arises whether or not the PSO equations can also be applied to other representations. For instance, in binary PSO [5] a variation is used by replacing the real valued position vector by a binary valued vector. But this is still a fix-sized vector representation only using a different data type for the components.

But what, if the size of your problem solution is unknown in advance? Imagine the clustering of an unknown set of data where you don't know the number of clusters *a priori*. Another example is the optimization of dynamic data structures, e.g., trees or graphs which can be represented by sets of tupels. For instance, in [14] the authors use an adjacency list as variable length representation to encode Dynamic Bayesian Networks. But their representation is specialized on this kind of domain.

Several works successfully use sets as representation for a genetic algorithm. For instance, Raidl and Julstrom use sets as representation in genetic algorithms to represent the edges of spanning trees [10]. In [7] the authors use a set oriented genetic algorithm to solve the knapsack problem.

In [9] the authors introduce a set-based PSO to solve set-based combinatorial problems (especially for determining RNA structures). They replaced the addition / subtraction operators by the typical set union / minus operations. But they also removed the scalar multiplication. To compensate the loss of diversity, they add random elements to particle positions. Weakly compared to the original PSO concept this means to use no scaling of the direction but to reset vector components randomly.

The concept proposed in this paper also replaces the position and velocity vectors of PSO by position and velocity sets, but intents to be more in the spirit of the original PSO as introduced by Kennedy and Eberhart [3]. All PSO operators are replaced by appropriate set operations and a special operator reduces possible bloating effects. Since the positions are sets of different cardinalities, this is a PSO with variable length representation. Because of the general way it is defined, it is a general purpose PSO with variable length representation. This set-based PSO is successfully applied to two real world data clustering problems.

This paper is organized as follows. Section 2 introduces the standard Particle Swarm Optimization algorithm. The definition of the set-based PSO is presented in sections 3 and 4. An operator to deal with bloating effects is given in section 5. In section 6 the conducted experiments with some results are presented. Finally, in section 7 some conclusions are drawn with an outlook to future works.

2 Particle Swarm Optimization

Particle Swarm Optimization (PSO), as introduced by Kennedy and Eberhart [3] [5], is an optimization algorithm modeling the flocking of birds flying around a peak in a landscape. In PSO the birds are substituted by particles and the peak in the landscape is the peak of a fitness function. The particles are flying through the search space forming flocks around peaks of fitness functions.

Let N_{dim} be the dimension of the problem (i.e., the dimension of the search space $\mathbb{R}^{N_{dim}}$), N_{part} the number of particles and $\mathcal{P}$ the set of particles $\mathcal{P} = \{P_1, ..., P_{N_{part}}\}$. Each particle $P_i = (x_i, v_i, l_i)$ has a current position in the search space ($x_i \in \mathbb{R}^{N_{dim}}$), a velocity ($v_i \in \mathbb{R}^{N_{dim}}$) and the locally (or personally) best found position in history, i.e., its own experience ($l_i \in \mathbb{R}^{N_{dim}}$).

In PSO, the set of particles $\mathcal{P}$ is initialized at time step $t = 0$ with randomly created particles $P_i^{(0)}$. The initial l_i are set to the corresponding initial x_i. Then, for each time step t, the next position $x_i^{(t+1)}$ and velocity $v_i^{(t+1)}$ of each particle $P_i^{(t)}$ are computed as shown in Eqns. (1) and (2).

$$v_i^{(t+1)} = w_I^{(t)} v_i^{(t)} + w_L r_1 (l_i^{(t)} - x_i^{(t)}) + w_N r_2 (n_i^{(t)} - x_i^{(t)}) \tag{1}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{2}$$

Here, r_1 and r_2 are random numbers in $[0,1]$. The personally best found position of the best neighbor particle at time t is denoted with $n_i^{(t)} \in \mathbb{R}^{N_{dim}}$.

As presented in [5] [6] [1] there are several possibilities to define the neighborhood of a particle. The global (*gbest*) PSO uses a star topology [4]. This way, the whole swarm is the neighborhood of each particle. For the local (*lbest*) PSO a ring topology is used [4]. The neighbors of a particle are only the ones at both sides up to a specified radius. A radius of r means r neighbors of the left and r neighbors of the right side.

The inertia weight $w_I^{(t)}$ determines the influence of the particle's own velocity, i.e., it represents the confidence of the particle to its own position (typically $w_I \in [0.1, 1.0]$). To yield a better convergence, this weight is decreased over time [11] [5] by an amount of a_I (typically $a_I = 0.001$). w_L is the influence of the locally best position found so far. The influence of the best particle of the neighborhood is denoted by w_N.

To avoid chaotic behavior, the new velocity $v_i^{(t+1)}$ is clamped to a pre-defined interval $[-V_{max}, +V_{max}]$. Also, the new position $x_i^{(t+1)}$ is clamped to the problem-specific range $[X_{min}, X_{max}]$. Often, X_{min} is set to $-X_{max}$.

The fitness of a particle is determined by a fitness function $F : \mathbb{R}^{N_{dim}} \rightarrow \mathbb{R}$. If the new position $x_i^{(t+1)}$ has a better fitness than the best solution found so far for particle P_i, it is stored in memory as shown in Eq. (3) (in case of minimization).

$$l_i^{(t+1)} = \begin{cases} x_i^{(t+1)} & , F(x_i^{(t+1)}) < F(l_i^{(t)}) \\ l_i^{(t)} & , otherwise \end{cases} \tag{3}$$

The best solution of the run is found at particle P_b with the best local solution l_b. Best solution l_b is always element of the set of all best local solutions $\{l_i\}, \forall i \in \{1, \cdots, N_{part}\}$. The best fitness value is $F(l_b) = \min_{i \in \{1, \cdots, N_{part}\}} \{F(l_i)\}$.

3 Abstract PSO

The standard PSO as described in section 2 uses vectors as representation. All operations used in Eqns. (1) and (2) are the scalar multiplication and vector addition / subtraction. Eqns. (4) and (5) define the PSO in a more abstract way. For simplification the weights and random numbers are combined by $\eta_L = w_L r_1$ and $\eta_N = w_N r_2$.

$$v_i^{(t+1)} = w_I^{(t)} \odot v_i^{(t)} \;\oplus\; \eta_L \odot (l_i^{(t)} \ominus x_i^{(t)}) \;\oplus\; \eta_N \odot (n_i^{(t)} \ominus x_i^{(t)}) \tag{4}$$

$$x_i^{(t+1)} = x_i^{(t)} \boxplus v_i^{(t+1)} \tag{5}$$

To use PSO for other representations, the operations $\oplus$, $\ominus$, $\odot$ and $\boxplus$ need to be redefined in an appropriate way.

4 Set-Based PSO

The set-based PSO replaces the position and velocity vectors by position and velocity sets containing domain-specific elements as defined by a set $\mathbb{D}$. To avoid confusion with the SetPSO approach introduced in [9], the set-based PSO proposed in this paper is called Set Swarm Optimization (SSO). Each particle $P_i = (x_i, v_i, l_i)$ has a current position in the search space ($x_i \in \mathcal{P}(\mathbb{D})$), a velocity ($v_i \in \mathcal{P}(\mathbb{D})$) and the personally best found position in history ($l_i \in \mathcal{P}(\mathbb{D})$). The scalars w_I, η_L, η_N are elements of the domain-specific set $\mathbb{S}$. More detailed examples are given in section 6.

4.1 Operations

As pointed out in section 3, the operations $\oplus$, $\ominus$, $\odot$ and $\boxplus$ need to be redefined for sets.

The addition $\oplus : \mathcal{P}(\mathbb{D}) \times \mathcal{P}(\mathbb{D}) \rightarrow \mathcal{P}(\mathbb{D})$ of velocity sets is realized simply as set union: $\forall v_i, v_j \in \mathcal{P}(\mathbb{D}) : (v_i, v_j) \mapsto v_i \cup v_j$.

The subtraction $\ominus : \mathcal{P}(\mathbb{D}) \times \mathcal{P}(\mathbb{D}) \rightarrow \mathcal{P}(\mathbb{D})$ of position sets is the set minus: $\forall x_i, x_j \in \mathcal{P}(\mathbb{D}) : (x_i, x_j) \mapsto x_i \setminus x_j$.

The scalar multiplication $\odot : \mathbb{S} \times \mathcal{P}(\mathbb{D}) \rightarrow \mathcal{P}(\mathbb{D})$ for velocity sets is applied element-wise using a domain-specific operator $\otimes_D : \mathbb{S} \times \mathbb{D} \rightarrow \mathbb{D}$ and is defined as: $\forall s \in \mathbb{S}, \forall v \in \mathcal{P}(\mathbb{D}) : (s, v) \mapsto \{s \otimes_D d_i | \forall d_i \in v\}$. Since $\otimes_D$ is applied independent to each element, this operation does not change the cardinality, i.e., $|s \odot v| = |v|$ for all $s \in \mathbb{S} \setminus \{s_0\}, v \in \mathcal{P}(\mathbb{D})$, whereby s_0 is the null element of scalars (if in existance). Note that, if you have a null element $d_0 \in \mathbb{D}$ and $s_0 \otimes_D d_i = d_0$ holds for all $d_i \in \mathbb{D}$, then $|s_0 \odot v| = 1$, since $|s_0 \odot v| = |\{s_0 \otimes_D d_i | \forall d_i \in v\}| = |\{s_0 \otimes_D d_1, \cdots, s_0 \otimes_D d_{|v|}\}| = |\{d_0, \cdots, d_0\}| = |\{d_0\}| = 1$.

The update addition $\boxplus : \mathcal{P}(\mathbb{D}) \times \mathcal{P}(\mathbb{D}) \rightarrow \mathcal{P}(\mathbb{D})$ of position and velocity sets is defined the same way as the $\oplus$ addition: $\forall x_i, v_i \in \mathcal{P}(\mathbb{D}) : (x_i, v_i) \mapsto x_i \cup v_i$.

4.2 Set Bloating

The operations in section 4.1 are able to produce a bloating effect. That means that the position and velocity sets get permanently more elements with ongoing iterations.

Velocity sets. Since the $\odot$ operation does not change the cardinality of the $w_I^{(t)} \odot v_i^{(t)}$ term (except $w_I^{(t)} = s_0$), the cardinality of the new velocity set $v_i^{(t+1)}$ is minimum the cardinality of the previous one ($v_i^{(t)}$). Additionally, all new elements created by the cognitive and social terms are unioned which increases the cardinality of $v_i^{(t+1)}$. Because the $\odot$ operator produces new elements by scaling them, the cognitive and social terms permanently enrich $v_i^{(t+1)}$ with new elements.

Position sets. Because the new velocity set $v_i^{(t+1)}$ bloats up and the $\boxplus$ operator is simply the set union, each new position set $x_i^{(t+1)}$ is the previous one ($x_i^{(t)}$) together with these new elements in $v_i^{(t+1)}$. Thus, the bloating effect from the velocity sets is passed through to the position sets.

To reduce these bloating effects a reduction operator $\mathcal{R}$ as defined in Eq. (6) is applied to the equations as presented in the next section.

$$\mathcal{R} : \mathcal{P}(\mathbb{D}) \rightarrow \mathcal{P}(\mathbb{D}) \tag{6}$$

4.3 Set Swarm Optimization

Now that all operations are defined, the SSO update equations can also be defined as in Eqns. (7) and (8). For simplicity most abstract operators are replaced by the used ones.

$$v_i^{(t+1)} = \mathcal{R}(\ w_I^{(t)} \odot v_i^{(t)} \ \cup \ \eta_L \odot (l_i^{(t)} \setminus x_i^{(t)}) \ \cup \ \eta_N \odot (n_i^{(t)} \setminus x_i^{(t)})\ ) \tag{7}$$

$$x_i^{(t+1)} = \mathcal{R}(\ x_i^{(t)} \cup v_i^{(t+1)}\ ) \tag{8}$$

Basically, SSO works like the standard PSO as described in section 2. A difference is the method of initialization. In standard PSO simply randomized position and velocity vectors are created. In SSO the position and velocity sets are created with a random number of randomized domain-specific elements, whereby the number of elements per set is between C_{min} and C_{max}. A randomized domain-specific element is created by a function $\mathcal{C}_{\mathbb{D}} :\rightarrow \mathbb{D}$.

A Set Swarm Optimization instance is characterized by a 4-tupel

$$SSO(\ \mathbb{D} \ , \ \mathbb{S} \ , \ \otimes_D \ , \ \mathcal{R} \ ) \tag{9}$$

and consists of 1) the set $\mathbb{D}$ of problem-specific objects, 2) the set $\mathbb{S}$ of problem-specific scalars, 3) the problem-specific scalar multiplication $\otimes_D$ and 4) a reduction operator $\mathcal{R}$. An example of a reduction operator $\mathcal{R}$ as used later on in the experiments is defined in the next section.

5 Reduction Operator

To overcome the bloating effects mentioned in section 4.2 a reduction operator is proposed. One idea to prevent bloating without influencing the search process to strong is to combine similar elements of a given position / velocity set. Here, this is achived by specifying a nonnegative threshold of nearness ε. All elements whose distance to each other is below this threshold are combined. In this sense a kind of clustering is performed according to a domain-specific element distance function δ as defined in Def. (1).

Definition 1 (Element Distance Function δ). *The distance function* $\delta : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{R}^+$ *computes the nonnegative distance between two domain-specific objects such that for all* $d_i, d_j \in \mathbb{D}$ *we have:*

1) $\delta(d_i, d_i) = 0$
2) $\delta(d_i, d_j) = 0 \ \Rightarrow \ d_i = d_j$

To combine similar elements a set-representative map ρ as defined in Def. (2) is used to compute or select a representative for a given set.

Definition 2 (Set-Representative Map ρ). *The map* $\rho : \mathcal{P}(\mathbb{D}) \rightarrow \mathbb{D}$ *computes or selects a representative for the given set such that for all* $D \in \mathcal{P}(\mathbb{D})$ *we have the single element identity:*

$$|D| = 1 \ \Rightarrow \ \rho(D) = \rho(\{d\}) = d$$

The reduction operator is denoted by $\mathcal{R}_{\varepsilon,\delta,\rho}$ and works as defined in Algorithm (1). In the first step all elements of a given set $D \in \mathcal{P}(\mathbb{D})$ are grouped together based on the element distance function δ. Then, in the second step all groups having elements in common are merged to ensure that all groups are pair-wise disjoint. Finally, in the third step for each group c_i a representative is determined by using a set-representative map ρ. If you want to deactivate the influence of the $\mathcal{R}_{\varepsilon,\delta,\rho}$ operator, you can set the ε

threshold to 0 ($\mathcal{R}_{0,\delta,\rho}$). The axioms of Defs. (1) and (2) ensure that $\mathcal{R}_{0,\delta,\rho}(D) = D$ holds for all $D \in \mathcal{P}(\mathbb{D})$. Additionally, the $\mathcal{R}_{\varepsilon,\delta,\rho}$ operator ensures that $\mathcal{R}_{\varepsilon,\delta,\rho}(\emptyset) = \emptyset$. In case the set-representative map ρ used is only defined over finite sets, the reduction operator $\mathcal{R}_{\varepsilon,\delta,\rho}$ is also restricted to finite sets. But this restriction can be disregarded, since in practice computers can not store an infinite number of elements.

Algorithm 1. Reduction Operator $\mathcal{R}_{\varepsilon,\delta,\rho}(D)$

```
// First step: Create groups $s_i$ of similar elements
$S \leftarrow \emptyset$
for all $d_i \in D$ do
    $s_i = \{d_i\} \cup \{d_j | \delta(d_i, d_j) \leq \varepsilon, \forall d_j \in D\}$
    $S \leftarrow S \cup \{s_i\}$
end for

// Second step: Merge all groups sharing same elements
$C \leftarrow S$
while $\exists s_i, s_j \in C : s_i \cap s_j \neq \emptyset$ do
    $C \leftarrow C \setminus \{s_i, s_j\}$
    $c' \leftarrow s_i \cup s_j$
    $C \leftarrow C \cup \{c'\}$
end while

// Third step: Compute representatives for all groups $c_i$
$P' \leftarrow \emptyset$
for all $c_i \in C$ do
    $P' \leftarrow P' \cup \{ \rho(c_i) \}$
end for

return $P'$
```

Two exemplary set-representative maps a) computing an average element and b) selecting the median element are defined in the following. These are also used in the experiments in section 6.

Definition 3 (Average Representative Map ρ_{avg}). *Let $D \in \mathcal{P}(\mathbb{D})$ be a finite set to be represented. Then, the map ρ_{avg} computes the average element over all elements $d_i \in D$:*

$$\rho_{avg}(D) = \frac{1}{|D|} \sum_{d_i \in D} d_i \tag{10}$$

ρ_{avg} is a set-representative map since it fulfills the single element identity:

$$\rho_{avg}(\{d\}) = \frac{1}{|\{d\}|} \sum_{d_i \in \{d\}} d_i = \frac{1}{1} d = d \tag{11}$$

Requirements: You can use ρ_{avg} only if you are able to provide an addition ($+ : \mathbb{D} \times \mathbb{D} \rightarrow \mathbb{D}$) and multiplication with a real number ($\cdot : \mathbb{R} \times \mathbb{D} \rightarrow \mathbb{D}$) for your domain $\mathbb{D}$. Furthermore, the 1 must be the identity element with respect to your $\cdot$ operation.

Definition 4 (Median Representative Map ρ_{med}). *Let $D \in \mathcal{P}(\mathbb{D})$ be a finite set to be represented. Then, the map ρ_{med} selects the median element out of all elements $d_i \in D$. For this, all elements are sorted according to an order operator $<_{\mathbb{D}}$: $\mathbb{D} \times \mathbb{D} \rightarrow \{0,1\}$. After sorting, the element in the middle position is picked out. Here, the median is thought to be one of the elements of the set D ($\rho_{med}(D) \in D$). If $(d_{(1)}, \cdots, d_{(|D|)})$ denotes the sorted list of all $d_i \in D$, then the median element is selected as follows:*

$$\rho_{med}(D) = \begin{cases} d_{(\frac{|D|+1}{2})} & \text{, if } |D| \text{ is odd} \\ d_{(\frac{|D|}{2})} & \text{, if } |D| \text{ is even} \end{cases} \tag{12}$$

ρ_{med} is a set-representative map since it fulfills the single element identity:

$$\rho_{med}(\{d\}) = d_{(\frac{|\{d\}|+1}{2})} = d_{(\frac{2}{2})} = d_{(1)} = d \tag{13}$$

Requirements: You can use ρ_{med} only if you are able to provide the order operator $<_{\mathbb{D}}$ for your domain $\mathbb{D}$.

Which one of the set-representative maps you choose depends on the operations you can provide. If you neither can provide the $+$ and $\cdot$ operations nor the $<_{\mathbb{D}}$ order operator, then you need to define your own set-representative map for your special domain $\mathbb{D}$.

6 Experiments

In this section, the results of two experiments are presented to show the capabilities of SSO. The used PSO parameters are the standard ones: $N_{part} = 20$, $w_L = 2$ and $w_N = 2$. The inertia weight w_I starts at 1 and is decreased by $a_I = 0.001$. For the local (*lbest*) versions, the radius r is set to 1. Validating the position and velocity sets is done according to the $[X_{min}, X_{max}]$ interval. The used ε values were determined in advance by a systematic parameter exploration. All experiments are averaged over 50 independent runs with 1000 iterations per run.

6.1 An Order Relation of n-Dimensional Vectors

In case the median representative map ρ_{med} as defined in Def. (4) is used, an order relation over $\mathbb{D}$ is needed. Assumed, the objects in $\mathbb{D}$ are n-dimensional vectors, i.e., $\mathbb{D} = V^n$. Then, the less order relation $<_{V^n}$ is a set containing sorted pairs of vectors: $<_{V^n} = V^n \times V^n$.

The order criterion used here is to order according to the first dimension and if the vector components of the first dimension are equal, order according to the second dimension and so on. In other words, the order is realized by ordering according to the first vector components which are unequal.

If $I = \{1, \cdots, n\} \subseteq \mathbb{N}$ is the index set of components of n-dimensional vectors, then the index set of all unequal vector components is $I_{\neq}(x,y) := \{\ i \mid x_i \neq y_i,\ \forall i \in I\ \} \subseteq I$ for all $x, y \in V^n$. Note that this order is applied to a set and not a multi-set. Thus, $I_{\neq}(x,y)$ will never be empty since in a set there will be no x, y which are equal.

The actual order relation as given in Eq. (14) orders two vectors x, y according to the unequal component with the minimal index ($\min I_{\neq}(x,y)$).

$$<_{V^n} := \{ \ (x,y) \ | \ x,y \in V^n, \ x_{\min I_{\neq}(x,y)} \ <_V \ y_{\min I_{\neq}(x,y)} \ \} \tag{14}$$

The $<_V$ operator is the less operator on single vector components. If $V = \mathbb{R}$, then $<_V = <_{\mathbb{R}} = <$ is the typical less operator as known for real numbers.

6.2 Data Clustering

The task of data clustering is to divide a set of data X into sub-sets (clusters) $C_i \subseteq X$ containing similar data. Used are the well-known datasets *Iris* and *Wisconsin Breast Cancer* (*WBC*) which were taken from the UCI Repository Of Machine Learning Databases [8]. The dataset *Iris* contains 3 clusters in 150 records with 4 numerical attributes. The dataset *WBC* contains 2 clusters in 683 records with 10 numerical attributes.

Let C denote the set of computed clusters $C_i \subseteq X$ and T the set of labels t of the real known classes of a dataset. Furthermore, let N_{ti} be the number of data items of class t within cluster C_i and N_i the size of cluster C_i, i.e., $N_i = |C_i|$.

The measure employed in this work is the F-measure known from information retrieval as shown in Eq. (15,left). It uses the purity of the considered cluster C_i with $Prec(t,C_i) = \frac{N_{ti}}{N_i}$, i.e., how strong belongs cluster C_i completely to class t. Furthermore, it considers how much of the data of class t are contained within cluster C_i with $Rec(t,C_i) = \frac{N_{ti}}{N_t}$ and N_t being the number of data in class t.

$$FM(t,C_i) = \frac{2 \cdot Prec(t,C_i) \cdot Rec(t,C_i)}{Prec(t,C_i) + Rec(t,C_i)} \ , \ FMeas(C) = \sum_{t \in T} \frac{N_t}{|X|} \max_{C_i \in C} \{FM(t,C_i)\} \tag{15}$$

The best situation is to have each cluster consisting completely of data of the same class t ($Prec(t,C_i) = 1$) and for each class t having all data placed in just one cluster ($Rec(t,C_i) = 1$). This measure is limited to $[0,1]$ and to be maximized. The overall F-measure value is determined as in Eq. (15,right).

Let the domain be real-valued vectors, i.e., $\mathbb{D} = \mathbb{R}^n$. Every position set represents a set of centroids. Computing the overall fitness of a set $D \in \mathcal{P}(\mathbb{D})$ involves two steps. Firstly, the set of clusters C is computed by interpreting each $d_i \in D$ as centroid. Each centroid d_i has its corresponding cluster C_i. All data items in X are assigned to these clusters, whereby a data item is assigned to the nearest cluster C_j according to Euclidean distance. Finally, the fitness $F(D)$ of a set $D \in \mathcal{P}(\mathbb{D})$ is the overall F-measure as shown in Eq. (16). Since the F-measure is to maximize and SSO is a minimizer, the logic is turned around.

$$F(D) = 1 - FMeas(C) \tag{16}$$

The fitness function F is to minimize.

The SSO used is

$$SSO(\ \mathbb{R}^n \ , \ \mathbb{R} \ , \ \cdot_{\mathbb{R}^n} \ , \ \mathcal{R}_{\varepsilon,\delta,\rho} \ ) \tag{17}$$

with $\cdot_{\mathbb{R}^n}$ being the scalar multiplication of real numbers with real vectors, $\varepsilon = 3$ for *Iris*, $\varepsilon = 8$ for *WBC* and $\delta = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$ $(a,b \in \mathbb{R}^n)$. For ρ experiments with both maps ρ_{avg} and ρ_{med} are conducted.

The dimensions of the used vectors are $n = 4$ (*Iris*) and $n = 10$ (*WBC*) according to the number of attributes. The ranges $[X_{min}, X_{max}]$ are set to $[0.1, 7.9]$ (*Iris*) and $[1, 10]$ (*WBC*) according to the minimum and maximum attribute values of the datasets. The number of initial elements is set to $C_{min} = 1$ and $C_{max} = 5$. Each randomized element is created by $\mathcal{C}_{\mathbb{R}^n} := U(X_{min}, X_{max})^n$.

The obtained results are given in Table 1. The methods with "median" in brackets refer to ρ_{med} and "average" refers to ρ_{avg}. As can be seen, the local SSO with the median representative map works best for both datasets. Compared to other methods as e.g. Data Swarm Clustering [13], *k*-means and Ant-based Clustering [2], the SSO approach produces better results. Even the worst SSO variants are better than the three other methods. The comparative values were taken from [13] [2] and were converted from "F-measure" values to "1 - F-measure" values, because the SSO method minimizes "1 - F-measure". Compared to the PSO clustering method ("PSO Clustering") as introduced in [12] similar results are obtained. The mean F-measure values are close, but SSO has a better standard deviation. Because the method introduced in [12] encodes all centroids in a sequence, the right numbers of clusters are given in advance. For the SSO approach this is not necessary.

Table 1. Performance of SSO on both datasets

	Iris	*WBC*
Global SSO (median)	0.034229 (± 0.013114)	0.023910 (± 0.001900)
Global SSO (average)	0.043435 (± 0.019289)	0.024099 (± 0.001048)
Local SSO (median)	**0.029916** (± 0.009529)	**0.022768** (± 0.001436)
Local SSO (average)	0.040805 (± 0.014850)	0.023631 (± 0.001163)
PSO Clustering	**0.029799** (± 0.032804)	**0.023465** (± 0.003343)
Data Swarm Clustering	0.169317 (± 0.073264)	0.314268 (± 0.062420)
Ant-based Clustering	0.183188 (± 0.014846)	0.032396 (± 0.001447)
***k*-means**	0.175479 (± 0.084866)	0.034175 (± 0.000000)

7 Conclusions and Future Works

As the results show it is possible to apply the PSO update equations to sets. But it can be neccessary to reduce a bloating effect caused by the interplay between the $\odot$ and $\cup$ operations.

In both experiments, the methods using ρ_{med} are better than those using ρ_{avg}. The local SSO with the median representative map produces the best results. Furthermore, the type of neighborhood topology (*gbest* or *lbest*) has only a small effect on the results. But for the *lbest* SSO only the standard radius was tested. Maybe the performance can be increased by choosing other radii or completely other topologies.

A still open question is the selection of the ε parameter. It should be examined whether this parameter can be determined or estimated for a given problem simplier than with parameter exploration. Maybe other reduction operators can be found without the need of a threshold.

References

1. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. Wiley, Chichester (2005)
2. Handl, J., Knowles, J., Dorigo, M.: On the performance of ant-based clustering. In: Proc. 3rd International Conference on Hybrid Intelligent Systems, pp. 204–213. IOS Press, Amsterdam
3. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center, Piscataway (1995)
4. Kennedy, J.: Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In: Proceedings of the 1999 Conference on Evolutionary Computation, vol. 3, pp. 1931–1938 (1999)
5. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
6. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, pp. 1671–1676 (2002)
7. Ku, S., Lee, B.: A Set-Oriented Genetic Algorithm and the Knapsack Problem. In: Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, May 27-30 (2001)
8. Murphy, P.M., Aha, D.W.: UCI Repository of machine learning databases, Irvine, CA: University of California, Department of Information and Computer Science, `http://www.ics.uci.edu/~mlearn/MLRepository.html`
9. Neethling, M., Engelbrecht, A.P.: Determining RNA Secondary Structure using Set-based Particle Swarm Optimization. In: Proceedings of the 2006 Congress on Evolutionary Computation, Vancouver, BC, July 16-21, pp. 1670–1677 (2006)
10. Raidl, G.R., Julstrom, B.A.: Edge-sets: An effective evolutionary coding of spanning trees. IEEE Transactions on Evolutionary Computation 7(3), 225–239 (2003)
11. Shi, Y.H., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: IEEE International Conference on Evolutionary Computation, Anchorage, Alaska (1998)
12. van der Merwe, D.W., Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Canberra, Australia, vol. 1, pp. 215–220 (2003)
13. Veenhuis, C., Köppen, M.: Data Swarm Clustering. In: Abraham, A., Grosan, C., Ramos, V. (eds.) Swarm Intelligence and Data Mining, Studies in Computational Intelligence. Springer, Germany (2006)
14. Xing-Chen, H., Zheng, Q., Lei, T., Li-Ping, S.: Research on Structure Learning of Dynamic Bayesian Networks by Particle Swarm Optimization. In: Proceedings of the First IEEE Symposium on Artificial Life, Honolulu, Hawaii, April 1-5 (2007)

Nature–Inspired Synthesis of Rational Protocols

Almudena Alcaide, Juan M.E. Tapiador,
Julio C. Hernandez-Castro, and Arturo Ribagorda

Computer Science Department, Carlos III University of Madrid
Avda. Universidad 30, 28911, Leganes, Madrid
{aalcaide,jestevez,jcesar,arturo}@inf.uc3m.es

Abstract. Rational cryptography is an emerging field which combines aspects traditionally related to security with concepts described in economic theoretical frameworks. For example, it applies game theory concepts to address security problems arising when executing cryptographic protocols. The aim is to replace the assumption of a worst–case attacker by the notion of rational agents that try to maximize their payoffs. In this work, we define a formal framework and a meta–heuristic technique for the automated synthesis of multi–party rational exchange security (M–RES) protocols. We provide experimental results for a simple scenario where a 3–party rational exchange protocol is automatically designed.

1 Introduction and Motivation

The *exchange problem* of how to design a general procedure according to which several parties can exchange items in a *fair* manner has attracted much attention throughout the years. Interest in this class of protocols stems from its importance in many applications where disputes among parties can occur, such as digital contract signing, certified e–mail, exchange of digital goods and payments, etc. Roughly, the property of fairness means that no party should reach the end of the protocol in a disadvantageous position. Still, there exist no protocol according to which a number of parties can exchange items in a fair manner exclusively by themselves, assuming that misbehaving parties take part in the protocol ([1].) As a result, all fair exchange protocols require a trusted third party (TTP) in order to preserve fairness during the exchange.

By contrast, *rational* exchange protocols do have the enormous advantage of not needing a TTP. Informally, a rational exchange protocol cannot provide fairness, but it ensures that rational (i.e. self–interested) parties would have no reason to deviate from the protocol, as misbehaving does not result beneficial. The work presented in this paper focuses on the automated design of this type of protocol (rational exchange protocols) in multi–party environments.

Next we motivate the need for this approach and introduce some related work.

Shortage of rational proposals. As it is only recently that rational exchange schemes have been considered as an alternative solution to the exchange problem, there are very few rational exchange protocols proposed in the literature (see e.g. [2], [3].)

Automated analytical tools versus automated designing tools. With regard to the formal analysis of security protocols, several automated tools have been presented over the years (see [4] for an excellent survey.) In every case the focus has always been on the validation of existing schemes. We intend to adopt a relatively novel approach integrating formal verification within the designing methodology.

Meta–heuristic search for automated protocol synthesis. It is clear that the number of possible protocols achieving a set of goals from a set of initial assumptions grows exponentially as the number of goals or the number of participant entities rise. In this context, a designing methodology based on meta–heuristic search appears to be a reasonable option. Hao, Clark and Jacob were the first in applying these techniques for the synthesis of protocols that are provable correct and satisfy certain security criteria ([5] and [6].) In their work, they present an automated tool, based on Simulated Annealing, which finds security protocols that achieve certain goals from a set of initial assumptions. In later work [7] the authors apply a different heuristic technique based on genetic algorithms for the synthesis of provably secure protocols. A similar approach is also adopted by Park et al. in [8] to the synthesis of cryptographic protocols for a fault–tolerant agent replication system.

1.1 Overview of Our Work

In this paper, we describe a framework for the automated synthesis of rational exchange protocols (Section 2). The practical implementation of this formalism results on an application designed to find multi–party rational exchange security (M–RES) protocols for specified scenarios. We will illustrate its practical application within a 3–party scenario (Section 3) where our heuristic technique finds rational solutions very efficiently.

2 Foundations

Simple linear structures such as vectors and matrices will be used to represent all aspects of a multi–party exchange problem. Prior to describing the model in detail, the following definition will serve to unify notation throughout the paper:

Definition 1 (Exchange protocol). *Given a set of entities $\mathcal{P} = \{P_1, \ldots, P_v\}$ and a set of items $\mathcal{O} = \{o_1, \ldots, o_m\}$, an exchange protocol Π consists of n steps, each denoted by* $(t)\ P_i \rightarrow P_j : o_{t_1}, \ldots, o_{t_{k_t}}$, *where:*

- $t = 1, \ldots, n$ *is the step number,*
- P_i*,* $P_j \in P$*,* $i \neq j$*, are the sender and receiver of the message, respectively.*
- $\{o_{t_1}, \ldots, o_{t_{k_t}}\} \subseteq \mathcal{O}$ *are the items* P_i *sends to* P_j*, subject to* P_i *owning those items at step* t *of the protocol.*

Note that this definition merely describes a series of messages being exchanged between participants so that, at the end of the protocol execution some entities would have lost control over some of their items as well as having gained access over new ones. Further along in the synthesis process, a fitness function will decide how good a protocol is in giving solution to a specific exchange problem.

2.1 Protocol Representation

Protocols described in Definition 1 will be represented by a series of matrices.

Protocol Matrix. A protocol Π is represented by a matrix $S^{\Pi} \in \mathcal{M}_{n\times(m+2)} = [s^{\Pi}_{i,j}]$ of integers, where each row is interpreted as a message in which the first two components identify the sender and the receiver of the message, respectively, and the rest of the row components represent the items being sent.

Although matrix S^{Π} represents the series of steps that participant entities have to take along a protocol execution, the actual real message content being sent at each step is subject to: (1) the sender entity holding the referred items at that point in the protocol run; and (2) those items being *accessible* to that sender. Different situations could derive in a *non–accessible* status of an item o_j for a particular entity P_i. For example, if an item o_j is encrypted and entity P_i does not hold the decryption key. Something similar happens if entity P_i is able to generate item o_j but it needs to gain access to other items in order to do so. In this case, item o_j must remain non–accessible until gaining control over the rest of the required tokens. During the protocol execution this kind of information, which is specific to the particular exchange problem at hand, will be captured in two additional matrices: a matrix $H(t)$ denoted *state matrix* and a matrix R, denoted *inter–dependency matrix* describing items' dependency relations. Both structures are described below.

State Matrix. For each step t in the protocol $t = 1, \ldots, n$, matrix $H(t) = [h_{i,j}(t)] \in \mathcal{M}_{v\times m}$ will capture the possessions of each party at the end of such a step. At the initial step ($t = 0$) the matrix will represent the possessions of each different entity prior to the exchange.

Inter-Dependency Matrix. A matrix $R = [r_{i,j}] \in \mathcal{M}_{(v\times m)\times(v\times m)}$ will capture the inter–dependency relations for each $h_{i,j} \in H$ for a given exchange problem. Two different types of dependency relations, *positive* and *negative*, can be expressed within the model as follows:

– Items o_i and o_j are positively related if when o_j is non–accessible then, gaining access to o_i implies gaining access to item o_j too.
– Items o_i and o_j are negatively related if when o_j is non–accessible, then receiving o_i implies making item o_i non–accessible.

Further and more complex dependency links may be represented in matrix R, involving several items. The only restriction imposed by this representation is that negative and positive relations between any two given elements cannot be simultaneously expressed.

Updating the State Matrix. Initially, a candidate solution consists of a protocol matrix S^{Π}, a state matrix $H(0)$, and an inter–dependency relation matrix R, specific to the exchange environment. As the protocol execution progresses, the state matrix H is updated according to the instructions given in the protocol and the positive and negative restrictions imposed by matrix R. At the end of the protocol execution $H(n)$ will reflect the possessions that each entity holds and also those items that each entity has lost control over. How good the protocol S^{Π} is in giving solution to a particular exchange problem will be decided by a fitness function.

2.2 Fitness Function

A fitness function is individually defined for each participant of the protocol to evaluate the gains obtained at each step of the scheme. The search will aim at finding exchanging schemes which maximize each individual fitness function.

Benefit Matrix. In our model, all participants assign every item involved in the exchange a particular value. Those values serve to represent each individual's set of requirements and are captured in matrix $B = [b_{i,j}] \in \mathcal{M}_{v \times m}$, denoted *benefit matrix.* A more formal description of this matrix is given next:

$$b_{i,j} = \begin{cases} 1 & \text{iff } P_i \text{ incurs cost when losing control over } o_j \\ 0 & \text{iff item } o_j \text{ is of no value to participant } P_i \\ -1 & \text{iff } P_i \text{ obtains benefit when losing control over } o_j \\ & (\textit{Via coalitions or incentives}) \\ > 1 & \text{iff item } o_j \text{ is required by participant } P_i \\ & \textit{(It represents the value that item } o_j \textit{ is worth to entity } P_i, \\ & \textit{if and only if, } o_j \textit{ becomes accessible to } P_i\textit{)} \end{cases} \tag{1}$$

Maximum and Minimum Benefit Values. The following criteria will serve to: (1) compute the maximum benefit that an entity can obtain in a single protocol run; and (2) compute the minimum benefit that each entity P_i will obtain, which satisfies its requirements.

– A maximum benefit value $\hat{b}_i$ represents the payoff obtained when the outcome of the protocol run is the most favorable for entity P_i. It is computed considering that the entity has gained access to all the required items, it has sent all items for which losing control over is beneficial and has kept all items for which sending represents a cost.
– Minimum benefit value $\bar{b}_i$ represents the minimum payoff that entity P_i would expect to obtain with the exchange. The minimum that a rational entity will consider as a satisfactory exchange is that in which the entity has gained access to all required items, has had to lose control over items for which sending represents a cost, and it does not possess any of the items for which relaying is beneficial.

We will now define a fitness function to compute participant "fitness" (i.e. benefit attained) after each step in the protocol, as well as global protocol fitness at the end of a protocol run.

Utilities. At each step t of the protocol (after updating state matrix $H(t)$ according to the transference of items), we shall refer to the gains achieved by a player so far as "utility" or "payoff" values. Each P_i's current utility at step t, $(0 \leq t \leq n)$ can be denoted as $u_i(t)$.

Differential Utilities. The *differential utility* du_i for a player P_i between steps t_1 and t_2, with $0 \leq t_1 \leq t_2 \leq n$, is defined as:

$$du_i(t_1, t_2) = u_i(t_2) - u_i(t_1) \tag{2}$$

Additionally, a *global differential utility* dU will measure the overall fitness of a protocol solution S^{Π}. This can be defined as the sum of the benefit attained by each participant at the end of the execution:

$$dU(S^{\Pi}) = \sum_{i=1}^{v} du_i(0, n) \tag{3}$$

2.3 On the Solution Space

Given the formalism just described, our goal will be to explore the space of all exchange protocols to find schemes which are:

1. **Feasible.** That is, the exchange described by the protocol solution S^{Π} would represent a feasible transference of the required items between each protocol participant and,
2. **Rational.** During a protocol execution, there may be steps at which players run into a temporarily "worse" state (i.e. $du_i(t, t+1) \leq 0$.) However, the relevant factors which ensure rationality of the scheme are:
   i. At the end of the protocol run, P_i must have gained enough differential utility. If $du_i(0, n) > 0$, the exchange is profitable to P_i, if $du_i(0, n) < 0$, the exchange is non–profitable to P_i and, when $du_i(0, n) = 0$ indicates that the exchange is of no use to P_i.
   ii. For each participant, the utility $u_i(n)$ must satisfy the minimum required by P_i $(u_i(n) \geq \bar{b}_i)$.
   iii. Finally, entities having attained their required minimum $\bar{b}_i$ in an intermediate step, should not be considered as active participants for the rest of the protocol. That is, entities achieving their goals must be forced to quit the protocol execution.

How Many Exchange Protocols Exist? As described in Section 2.1, a protocol is represented by a matrix $S^{\Pi} \in \mathcal{M}_{n \times (m+2)}$. An estimate of the total number of possible exchange protocols can then be given by:

$$\mathcal{O}\left(\frac{v!}{(v-2)!} 2^{nm}\right) = \mathcal{O}\big(v(v-1)2^{nm}\big) = \mathcal{O}(v^2 2^{nm}) \tag{4}$$

where n is the number of protocol steps, v is the number of entities and m is the number of tokens involved in the exchange.

It is difficult to determine how many of these protocols represent *feasible* solutions to a specific exchange problem. Even more challenging is to estimate how many of those feasible solutions represent a *rational* exchange. An heuristic search based on Simulated Annealing will assist in finding those protocol designs within the solution space of a given multi–party exchange problem, which satisfy the above conditions of feasibility and rationality.

3 Automated Synthesis of a 3–RES Protocol

For the purpose of this paper, we will focus on a particular 3–entity exchange problem. For every participant entity we will provide a series of initial assumptions and goals which will be represented using the matrices described in Section 2.

3.1 A 3–RES Problem

Initial assumptions and other aspects of the particular exchanging problem are formalized as follows:

1. The specific exchange problem will consist of an entity P_0 which aims to collect a series of electronic items from entities P_1 and P_2, delivering the appropriate tokens in return. All entities, P_0, P_1 and P_2, are considered to be rational (aimed to maximize their payoffs). The following items are involved in the scheme:
   - o_0: Request token issued by P_0 containing a description of the item that P_0 requires from P_1.
   - o_1: Request token issued by P_0 containing a description of the item that P_0 requires from P_2.
   - o_2: Return token issued by P_0 for P_1 in return for o_4.
   - o_3: Return token issued by P_0 for P_2 in return for o_5.
   - o_4: Customized item issued by entity P_1 as specified by P_0 in o_0.
   - o_5: Customized item issued by entity P_2 as specified by P_0 in o_1.
2. None of the collected items in isolation is of any value to entity P_0. In other words, P_0 is interested in collecting all (i.e. o_4 and o_5) or none of these items.
3. Entities must choose an arbitrary positive integer greater than one, for each one of their required items. These values will represent the payoff associated to gaining access to such items.
4. Finally, the nature of these items is such that their utilities only become available when the corresponding tokens are delivered in return. Although this restriction seems hard, there are a few real life examples where items are of this nature. For example, P_0 could be a user trying to book a holiday package consisting of accommodation, flights and tickets for a local tourist attraction. User P_0 needs either all or none of the required items and, additionally, no item becomes available unless the providers of the required services have received payment.

3.2 Search Technique and Parameterization

Simulated Annealing (SA) [9] will be used as search technique. The basic algorithm has been slightly modified to stop when the first rational exchange protocol which satisfies the requirements is found. (This can be done by previously computing the minimum required global fitness.)

A simple random mutation mechanism is employed as move operator. Given a candidate solution (specified by a protocol matrix S^{Π}), a neighbor is obtained by randomly modifying a percentage of its elements. We will refer to the amount of elements mutated in the matrix as the *moving rate*. The different moving rates considered in the experimental work are: 1%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70% and 80%.

The acceptance criterion in SA is given by:

$$s' \text{ is accepted if } f(s') - f(s) > T_i \ln u \tag{5}$$

where s and s' are, respectively, the current and mutated solutions, T_i is the current temperature, and u is a random number uniformly generated in $[0, 1]$.

At each cycle, the temperature is geometrically decreased by:

$$T_{i+1} = \alpha T_i \tag{6}$$

$0 < \alpha < 1$ being the cooling factor.

Note that, after m cycles the temperature is $T_m = \alpha^m T_0$ where T_0 is the initial temperature. For T_m to be very close to 0 (say $\epsilon = 10^{-6}$) after m cycles, a cooling rate of:

$$\alpha = \left(\frac{\epsilon}{T_0}\right)^{\frac{1}{m}} \tag{7}$$

is needed.

For our experimental work, these SA parameters have been adjusted according to the definition of two different profiles. Both, **profiles (I) and (II)**, will satisfy the following property: in the first cycle, the probability of accepting a bad move which decreases the global fitness value by just one unit will be approximately 0.5. Moreover, in profile (I), by half the total number of cycles, the probability of accepting a bad move which decreases the global fitness value by more than one unit will be almost 0. So from exactly half the total number of cycles onwards, the search will behave as a pure hill climbing (HC.) By contrast, in profile (II), the probability of accepting a bad move which decreases the global fitness value by more than one unit will be almost 0 by one quarter of the total number of cycles. In this case, from exactly one forth of the total number of cycles onwards, the search will behave as a pure hill climbing.

Other parameters for our specific problem are the following. There are 3 parties in the exchange which exchange 6 items according to the scenario described in Section 3.1. The maximum allowed number of messages per protocol is set to 10, with each message consisting of at most 6 items. Note that with these parameters the search space (possible number of protocols) is $\mathcal{O}(2^{63})$, according to expression (4).

Table 1. Rate of success (RS) and average number of protocols evaluated (NPE) per trial and moving rate (MR). Results estimated over 500 trials.

	Hill Climbing		**SA (Profile I)**		**SA (Profile II)**	
MR	RS	Avg. NPE	RS	Avg. NPE	RS	Avg. NPE
0.01	64.8%	108348	79.4%	82434	71.6%	90830
0.05	93.6%	47068	99.6%	27637	97.2%	29765
0.1	97.6%	36464	99.6%	24833	99.4%	23879
0.2	98.2%	35274	99.2%	32709	99%	29981
0.3	90.4%	59144	98.4%	47924	95%	50588
0.4	56.6%	42754	75.8%	67340	68.4%	55958
0.5	18.6%	26387	30.20%	56718	23.8%	44678
0.6	5%	14311	11.2%	40484	7%	29673
0.7	1.4%	11281	5%	37466	3.2%	24909
0.8	2%	10047	3.6%	34830	1%	24535

(a)

Random Search	
RS	Avg. NPE
2.0%	19823
6.4%	96975
12.4%	186519
23.0%	355611

(b)

3.3 Results

Extensive experimentation has demonstrated that around 200 cycles with 1000 moves in the SA inner loop are sufficient to reach solutions in reasonable time (around 1 or 2 minutes.)

Table 1(a) shows the results obtained for the two SA profiles and different moving rates (column MR). The rate of success (column RS) represents the percentage of executions attaining a feasible rational protocol over 500 trials. The average number of protocols evaluated is indicated in column Avg. NPE. Finally, both SA profiles (I and II) are compared with the results obtained when applying a classic Hill Climbing algorithm (HC.)

In both SA profiles, the best results are obtained with a moving rate of 0.1, which results in around a 99.5% of success (i.e. almost every execution produces a valid solution) by evaluating approximately 24500 protocols. These numbers imply synthesizing a protocol for this scenario in less than 1 minute in a common laptop. The success rate for slightly lower or higher mutation rates are similar, though the number of total candidates evaluated before reaching a solution grows considerably, thus resulting in a more inefficient search. As expected, higher mutation rates transforms the search in an almost random procedure with fewer chances to succeed. Further comparatives are shown in Table 1(b) where a random search is applied to resolve the same problem.

All in all, the best rates of success are systematically achieved by SA. Even though a simple HC technique attains very good solutions too, the average number of protocols evaluated per trial serves as an experimental proof of efficiency in favor of a more sophisticated heuristic based on SA. Furthermore, our preliminary experimentation indicates that this is certainly the case in more complex exchange scenarios. Finally, as for a pure random search, the numbers are several orders of magnitude below the results obtained by any of the other two techniques.

$$\begin{tikzcd}
P_0 \arrow[r, "o_0{,}o_1"] & P_1 \arrow[d, "o_1{,}o_4"] \\
& P_2 \arrow[ul, "o_4{,}o_5"]
\end{tikzcd}
\qquad
\begin{tikzcd}
P_0 \arrow[r, dotted, "o_2{,}o_3"] & P_1 \arrow[d, dotted, "o_3"] \\
& P_2
\end{tikzcd}$$

(a)

(1) $P_0 \rightarrow P_1 : \{o_0, o_1\}_{K_{P_1}}$
(2) $P_1 \rightarrow P_2 : \{\{o_1, o_4\}_{K_{P_1}^{-1}}\}_{K_{P_2}}$
(3) $P_2 \rightarrow P_0 : \{o_4, o_5\}_{K_{P_2}^{-1}}$
(4) $P_0 \rightarrow P_1 : \{o_2, o_3\}_{K_{P_1}}$
(5) $P_1 \rightarrow P_2 : \{\{o_3\}_{K_{P_1}^{-1}}\}_{K_{P_2}}$

(b)

Fig. 1. A synthesized 3–entity rational exchange protocol. The protocol runs in the two phases illustrated on the right.

3.4 An Example of 3–RES Protocol

Figure 1(a) shows an example of a synthesized protocol for the problem described in Section 3.1. Further security refinements are applied to each message resulting in the scheme shown in Figure 1(b) (K_{P_i} and $K_{P_i}^{-1}$ denote P_i's public and private keys, respectively.)

The 3–RES protocol synthesized using our proposed approach can be formally proven rational using techniques based on game theory and backward induction (see [10].) Informally, here are some aspects of the formalism which ensure that the scheme is a feasible rational solution satisfying all participants' sets of requirements:

- **From entity's P_0 point of view.** As stated in the initial assumptions, items o_4 and o_5 are of no use to entity P_0 until the corresponding return items o_2 and o_3 have reached entities P_1 and P_2, respectively. To this regard, and since entity P_0 requires either all or none of these items, entity P_0 is *rationally* forced to perform step (4) of the protocol.
- **From entities P_1 and P_2 point of view.** Again, the assumption of P_0 requiring either all or none of the items forces (rationally) entity P_1 to send messages (2) and (5) and entity P_2 to send message (3).

Therefore, no entity would unilaterally deviate from the 3–RES protocol as they could not obtain better utility values in doing so. The scheme is then a rational solution.

4 Conclusions

Traditionally, automated tools have always been applied to the analysis and verification of existing security protocols. In this paper we have adopted a new approach, ensuring rationality as part of the automated design of an exchange scheme.

For the purposes of this work, we have designed and implemented a 3–RES search algorithm based on Simulated Annealing. Moreover, the formal foundations of our methodology ensure high levels of flexibility and scalability for any multi–party rational exchange problem.

References

1. Pagnia, H., Gärtner, F.: On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt University of Technology, Department of Computer Science (1999)
2. Buttyán, L.: Building blocks for secure services: Authenticated key transport and rational exchange protocols. Technical report, Swiss Federal Institute of Technology. Lausanne (EPFL), Ph.D. Thesis No. 2511 (2001)
3. Syverson, P.: Weakly secret bit commitment: Applications to lotteries and fair exchange. In: Proceedings of the 11th IEEE Computer Security Foundations Workshop, pp. 2–13 (1998)
4. Kremer, S.: Formal analysis of optimistic fair exchange protocols. Technical report, Université Libre de Bruxelles. Faculté de Sciences Ph.D. Thesis (2003)
5. Clark, J., Jacob, J.: Protocols are programs too: the meta-heuristic search for security protocols. Information and Software Technology 43, 891–904 (2001)
6. Chen, H., Clark, J., Jacob, J.: Automatic design of security protocols. Computational Intelligence 20, 503–516 (2004); Special Issue on Evolutionary Computing in Cryptography and Security
7. Clark, J., Jacob, J.: Searching for a solution: engineering tradeoffs and the evolution of provably secure protocols. In: Proceedings IEEE Symposium on Security and Privacy (2002)
8. Park, K., Hong, C.: Cryptographic protocol design concept with genetic algorithms. In: KES (2), pp. 483–489 (2005)
9. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science 220, 671–680 (1983)
10. Alcaide, A., Estévez-Tapiador, J., Hernandez Castro, J., Ribagorda, A.: A multiparty rational exchange protocol. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2007, Part I. LNCS, vol. 4805, pp. 42–43. Springer, Heidelberg (2007)

Optimizing Real-Time Ordered-Data Broadcasts in Pervasive Environments Using Evolution Strategy

Rinku Dewri, Darrell Whitley, Indrajit Ray, and Indrakshi Ray

Colorado State University, Fort Collins, CO 80523, USA
`{rinku,whitley,indrajit,iray}@cs.colostate.edu`

Abstract. We consider the problem of real-time data broadcast scheduling in pervasive systems with soft deadlines and constraints on the order in which data items should be broadcast to be useful. The broadcast schedule needs to be generated to provide a certain level of quality of service. Thus, the real-time scheduler has to effectively trade-off between its running time and the quality of schedules generated. We use an evolution strategy to solve the problem. The variants tested includes $(1+\lambda)$-ES, $(1, \lambda)$-ES, and a $(2+1)$-ES with a modified Syswerda recombination operator, as well as a genetic algorithm.

Keywords: Data broadcasting, Scheduling, Evolution strategy.

1 Introduction

Many pervasive application domains involve clients interested in groups of related data items that can be processed one at a time following some order. Consider the following example – a traffic management system in a large city. The system gathers current traffic information using sensors and disseminates it to drivers in real time on demand. Drivers use smart GPS navigation units to request road conditions so that they can be routed and re-routed along the best possible roads. The GPS unit periodically requests traffic information for the roads that are still remaining in its route plan. The traffic service needs to provide the road conditions in the same order that the roads are in the route plan. That is, if the current route plan is $\langle r_1, r_2, r_3, r_4 \rangle$, where r_i is a road identifier, then the traffic service should provide the information as $\langle rc_1, rc_2, rc_3, rc_4 \rangle$, where rc_i is the road condition for r_i. A scheduling problem occurs in such an application when the number of data access requests is larger than the bandwidth capacity of the server.

Various soft deadlines may also be imposed on the requested data items, which if not served within a specific time window may result in the data item having near zero utility when finally received. For example, from the time that a driver makes a request for traffic information on road r_i to the time the monitoring service reports back with a gridlock on r_i, the driver may already have missed a more convenient exit for an alternate route. Given the resource limitations, it is not always possible that the time constraints of every incoming request be satisfied. Thus, a broadcast schedule is sought which can serve clients with as much utility as possible. The scheduling problem is more difficult in a real-time setting where generation of a high utility schedule has to respect run time constraints as well.

In this paper, we first propose an utility accrual method for data requests involving constraints on the order of the requested data items. Second, we define a modified version of the Syswerda recombination operator for use in methods with recombination. Finally, the utility function is used by an evolution strategy based schedule optimizer to evaluate the effectiveness of the schedules. We pay particular attention to the run time constraint of the scheduler and argue that simple variants of evolution strategy can be employed to satisfy this requirement.

The remainder of the paper is organized as follows. Section 2 outlines the related work. Section 3 presents the utility function and a formal statement of the problem. Section 4 presents the specifics of the evolution strategy method. The experimental setup is presented in Sect. 5. Discussion on the results obtained are presented in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Related Work

Several shortcomings of using a strict deadline based system are discussed by Ravindran et al. [1]. Jensen et al. point out that real-time systems are usually associated with a value based model which can be expressed as a function of time [2]. They introduce the idea of $Time - Utility\ Functions$ to capture the semantics of soft time constraints specifying utility as a function of completion time. An attempt to understand the benefit of utility values in hard deadline scheduling algorithms is made by Buttazzo et al. [3]. Wu et al. study a task scheduling problem where utility is considered a function of the start time of the task [4].

Ordered queries have been studied in the mobile computing paradigm. Chehadeh et al. take into consideration object graphs to cluster related objects close to one another in the broadcast channel [5]. Hurson et al. propose an extension for multiple broadcast channels [6]. Huang et al. show that several cases of the problem of broadcasting related data is NP-hard and propose a genetic algorithm to address the problem [7].

Our approach to the problem differs in the introduction of an utility metric to evaluate the goodness of schedules and uses evolution strategy as a fast and effective method to obtain high utility schedules.

3 Broadcast Scheduling

Data broadcasting is an efficient approach to address data requests, particularly when similar requests are received from a large user community. *Push-based* architectures broadcast commonly accessed data at regular intervals. *On-demand* architectures allow the clients to send their requests to the server. Certain requests have an added requirement of data items being served in a particular order. Further, particular emphasis has to be paid to the time criticality and utility of a served request in an on-demand architecture.

3.1 Broadcast Model

Clients use an uplink channel to a data provider to request various data items served by the provider. Each request Q_j takes the form of a tuple $\langle D_j, R_j \rangle$, where R_j is the response time within which the requesting client expects the first data item from the ordered set D_j, hereafter called a *data group*. The assumption that processing at the client end can start as soon as the first data item in the group is received is implicit in this context. Nonetheless, a client must receive all requested data items for the processing to complete. The data provider reads the requests from a queue and invokes a scheduler to determine the order in which the requests are to be served. It is important to note that new requests arrive frequently into the queue which makes the task of the scheduler dynamic in nature. A request is considered to be *"fully served"* when the last data item in the requested data group is retrieved, otherwise it is considered *"partially served"*.

The scheduler is invoked every time a new request is received. At each instance, the scheduler first determines the requests that will be fully served by the current broadcast and removes them from the request queue. For the remaining requests, the data items required to serve them are determined and a schedule is generated. The scheduler makes a "best effort" at generating a schedule that respects the response time requirements of the clients.

3.2 Utility Metric

In order to facilitate soft deadlines, we make the assumption that the utility of a data item received by a client decreases exponentially if not received within the expected response time. For request Q_j arriving at time T_j and involving the data group $D_j = \{d_{1j}, d_{2j}, \ldots, d_{N_j j}\}$, let $t_{1j}, t_{2j}, \ldots, t_{N_j j}$ be the time when the respective data items are retrieved by the client. The utility generated by serving the first data item is given as,

$$u_j[t_{1j}] = \begin{cases} 1 & , t_{1j} - T_j \leq R_j \\ e^{-\alpha(t_{1j} - T_j - R_j)} & , t_{1j} - T_j > R_j \end{cases} \tag{1}$$

The utility from the subsequent items is then given as, for $i = 2, \ldots, N_j$ and inter-item response time R_T,

$$u_j[t_{ij}] = \begin{cases} u_j[t_{(i-1)j}] & , t_{ij} - t_{(i-1)j} \leq R_T \\ u_j[t_{(i-1)j}] e^{-\alpha(t_{ij} - t_{(i-1)j} - R_T)} & , t_{ij} - t_{(i-1)j} > R_T \end{cases} \tag{2}$$

The utility of a data item for a client decays by half as a function of the response time, i.e. $\alpha = \ln 0.5/R$, where $R = R_j$ for the first data item in the requested group and $R = R_T$ for any subsequent data item.

If all data items are broadcast in a timely manner, a maximum utility of 1 will be generated by each data item. However, when a data item's broadcast time exceeds its expected response time, not only will its utility drop, it will also influence the maximum utility that can be obtained from subsequent items. We then say that the utility generated by serving the request is given by the utility generated at the last item of the data group, or $U_j = u_j[t_{N_j j}]$. The quality of service desired in an application domain can be directly specified as a fraction of the utility derived from serving the requests. For a schedule S, generated to serve the requests $Q_1, Q_2, \ldots, Q_M$ in the queue, the utility generated by the schedule is the aggregation of the utilities for the requests in the queue, given as,

$$U_S = \sum_{k=1}^{M} U_k \tag{3}$$

3.3 Problem Statement

A data source $D = \{D_1, D_2, \ldots, D_N\}$ is a set of N ordered sets (or data groups), where $D_j = \{d_{1j}, d_{2j}, \ldots, d_{N_j j}\}$ with N_j being the cardinality of D_j and $j = 1, \ldots, N$. All data items d_{ij} are assumed to be unique and are of equal size d_{size}. A request queue at any instance is a dynamic queue Q with entries Q_j of the form $\langle D_j, R_j \rangle$, $D_j \in D$ and $R_j \geq 0$. At an instance t_{curr}, let $Q_1, Q_2, \ldots, Q_M$ be the entries in Q. We define the notation $Wait[Q_j]$ to denote the data item that the request Q_j is currently waiting for. Further, we define $Rem[Q_j]$ as the ordered subset of data items that has been requested in Q_j but not yet received, i.e. $Rem[Q_j] \subseteq D_j$. A schedule is a total ordering of the elements in the multi-set $\bigcup_{j=1,\ldots,M} Rem[Q_j]$.

The time instance at which a particular data item from the schedule starts to be broadcast is dependent on the bandwidth of the broadcast channel. A broadcast channel of bandwidth b can transmit b *data unit* per *time unit*. If t_{ready} is the ready time of the channel (maximum of t_{curr} and the end time of current broadcast), then for the schedule $d_{i_1 j_1} < d_{i_2 j_2} < \ldots < d_{i_P j_P}$, the data item $d_{i_k j_k}$ can be retrieved by an interested client at time instance $t_{i_k j_k} = t_{ready} + [(k-1)d_{size}/b]$. All requests Q_j in Q with $Wait[Q_j] = d_{i_k j_k}$ are then partially served, i.e. $t_{i_k j}$ for such requests is set to $t_{i_k j_k}$, and $Rem[Q_j]$ is changed to $Rem[Q_j] - \{d_{i_k j_k}\}$. The request is fully served when $Rem[Q_j] = \phi$.

At each scheduling instance, the problem then is to find a schedule with maximum utility given by (3).

4 Solution Methodology

Evolution Strategies (ES) [8] are typically expressed by the μ and λ parameters. In the $(\mu + \lambda)$-ES, μ best of the combined parent and offspring generations are retained using truncation selection. In the (μ, λ)-ES variant, the μ best of the λ offspring replace the parents. We restrict our focus to simple stochastic local search variants by setting $\mu = 1$. We also experiment with a $(2+1)$-ES with recombination and a simple genetic algorithm (GA).

A schedule can be represented by a permutation of the data items currently in the $Rem[\cdot]$ sets of requests. Since the same data item may be present multiple times in this permutation, a request identifier is attached to every data item. For the requests $Q_1, Q_2, \ldots, Q_M$ currently in the request queue, the data items in $Rem[Q_j]$ of a request are identified by the tuples $\langle j, d_{i_k j} \rangle$, where $d_{i_k j} \in Rem[Q_j]$. The first component of a tuple identifies the request for which it exists in the permutation, and the second component identifies the data item number. The request identifier is only used in the permutation and is not part of the broadcast for the data item. Hence, if a request is waiting on a data item, say $d_{i_{k_1} j_1}$, then upon broadcast it will be retrieved by the request irrespective of the request identifier attached to the data item in the tuple, j_1 in this case.

The simplest ES methods employ a mutation operator only, which implies a low overhead on the running time of the scheduler in a dynamic setting. "Shift" mutation selects two random positions in a permutation and the element at the first chosen position is removed and inserted at the second chosen position. The second random position is chosen in a way such that the ordering constraints for the data item in the tuple is preserved.

Recombination for the $(2+1)$-ES and the GA is achieved by a modified version of the Syswerda order-based crossover operator [9]. Syswerda's operator chooses random positions on a parent for exchange with the other. However, doing so can disrupt the ordering constraints of the permutation on the offspring (Fig. 1). The modified operator, called the *constrained-Syswerda* operator, eliminates this problem by restricting the choice of positions to a random contiguous block instead. One can prove this modification always yields feasible schedules.

```
Valid Parent 1        :  a b c d e f g h i j
Valid Parent 2        :  a d e b c h i f g j
Cross Positions       :  *   *   *
Invalid Offspring     :  a b e d c f g h i j
```

Fig. 1. A counter example for the Syswerda order-based crossover operator. Ordering constraints are $\{a, b, c\}$, $\{d, e, f, g\}$ and $\{h, i, j\}$; constraint $\{d, e, f, g\}$ is violated in the offspring.

5 Experimental Setup

Due to the non-availability of a standard test dataset in the domain, the data used in our experiments is generated using well known distributions that are known to capture the dynamics of a public data access system [10]. The dataset contains 10000 requests generated using a Poisson distribution with an arrival rate of 3 requests per second. Each request consists of an arrival time, data group number, and an expected response time for the first data item in the group. We consider 100 different data groups, the number of data items in each drawn from an exponential distribution with rate parameter 10. The bandwidth for the broadcast channel is set at $200KB/s$.

The data groups requested are assumed to follow the commonly used *Zipf* distribution [10] with the characterizing exponent of 0.8. Under this assumption, the first data group becomes the most requested one, while the last one is the least requested. Broadcast schedules can be heavily affected by the size of the most requested data group. Thus, we consider two different assignments: INC – most requested data group has the smallest number of data items, and DEC – most requested data group is the largest in size [11,12]. Each data item is of size $50KB$.

Expected response times for the first data item are assigned from a normal distribution with mean 60s and standard deviation 20s. The particular settings of these parameters in our experiment results in expected response times to be generated in the range of $[0, 120]$s with a probability of 0.997. A zero value generated using this distribution implies that the request is expected to be served immediately. Any negative value is replaced by zero. Response times for intermediate data items is set at 1s.

For different variants of the $(1 + \lambda)$-ES and $(1, \lambda)$-ES, we fixed the maximum number of function evaluations to 15000. The run time of a scheduling instance is around 0.01s (on a 2×2.66 GHz Xeon running Fedora Core 8 with 2GB memory) with this setting. The number of function evaluations is fixed at 5000 for the $(2 + 1)$-ES. The parameters for the GA is set as follows: population size $=$ 100, number of function evaluations $=$ 15000, crossover probability $=$ 0.8, mutation probability $=$ 0.05, and 2-tournament selection.

6 Empirical Results

We present the results obtained from different variants of the ES on the two different data group assignments – INC and DEC. The results are averaged for 20 runs for each variant. Figure 2 shows the percentage global utility obtained by running $(1+\lambda)$-ES with different λ values. Recall that each request can have a maximum utility of 1. The percentage global utility is thus computed from the fraction of this maximum utility generated in the requests in the data set. For the INC type assignment, a $(1 + 1)$-ES yields an acceptably high ($> 90\%$) utility level. Given the bandwidth limit of $200KB/s$, up to 4 data items can be transmitted in a single time unit. In the case of an INC type assignment, this can serve at least 4 different requests for the frequently requested data groups. Note that the INC assignment has the most requested data group as the smallest in size, which is one data item in our experimental data set.

The DEC type assignment has 20 data items in the most frequently requested data group. Further, the *Zipf* distribution makes the larger data groups more often requested than the smaller ones. The $200KB/s$ bandwidth poses a hard bottleneck in this situation. Quite often, different requests for the same data group arrive at different times prohibiting the $Wait[\cdot]$ value of those requests to be the same. The $(1 + 1)$-ES fails to provide the same level of performance as it does for the INC assignment. Increasing the sampling rate λ to $3, 5$ and 10 show improvements up to 80% utility levels. Although increasing the number of generations to 2000 improved the utility level up to 86%, the number of function evaluations $(2000 \times 10 = 20000)$ exceeded the maximum set limit of 15000.

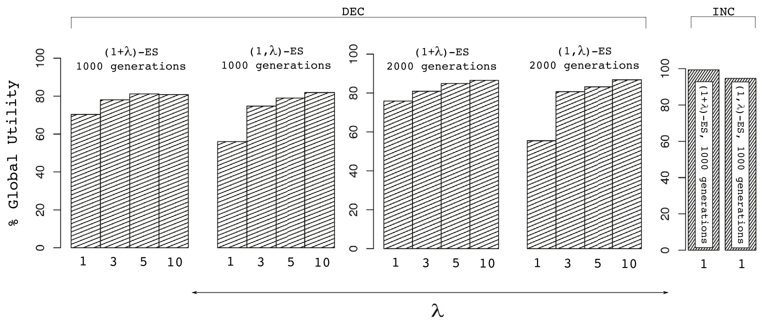


Fig. 2. Percentage global utility for the different λ values in the ES. For the *INC* assignment, a high utility level is obtainable with $\lambda = 1$. For the *DEC* assignment, increasing the sampling rate λ and the number of generations results in improvement of the global utility.

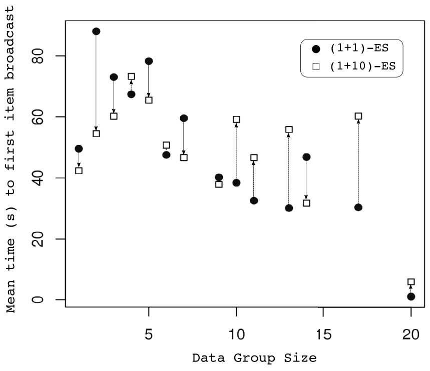


Fig. 3. Mean difference between time of request arrival and time of first data item broadcast for data groups of different sizes in the *DEC* assignment

The primary difference between the schedule utilities generated by $(1+1)$-ES for 1000 generations and $(1+10)$-ES for 2000 generations is attributable to the latency that the scheduler puts in between the time of arrival of a request and the time when the first data item for the request is broadcast. Figure 3 shows the mean values of this latency for data groups of different sizes. The $(1+10)$-ES maintains a higher mean latency for the larger data groups, whereas the $(1+1)$-ES maintains a higher value for the smaller data groups. Given that most requests are for the larger data groups in the *DEC* assignment, postponing the first data item broadcast for such requests provides gaps in the schedule to serve pending (or partially served) requests. Recall that the expected response time is the highest for the first data item (between 0 and 120s). Once a client has received the first data item, the expected response time drops to $R_T = 1s$. Hence, by delaying the first data item broadcast for the most requested data groups, the $(1+10)$-ES maintains a better flexibility in serving pending requests.

For the experimental data set, a simple (1 + 1)-ES for 1000 generations is sufficient when the QoS requirement is not too high (< 75%). For the case when the utility requirement is higher, a higher sampling rate is desired. However, note that results obtained from running the different variants for 2000 generations (more function evaluations) do not always yield a high difference in the utility levels as compared to those from running the same variants for 1000 generations. This is observed in both the *comma* and *plus* variants of the ES.

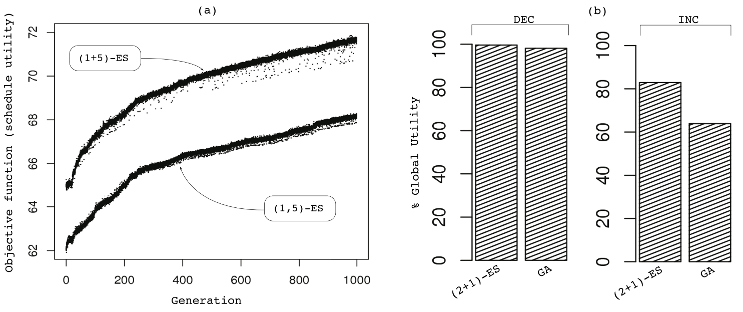


Fig. 4. (a) Schedule utility progress during the 5000^{th} scheduling instance with *DEC*. Utilities are also plotted for 100 points sampled around the parent of every generation. (b) Percentage global utility for $(2 + 1)$-ES and a genetic algorithm.

Figure 4a shows the changes in the objective function (schedule utility given by (3)) value during the scheduling instance when the 5000^{th} request arrives. The scheduler is working with *DEC* and $\lambda = 5$. Further, at each iteration, the plot shows the utilities of 100 randomly sampled points (using shift mutation) near the current parent. Note that the rise in the utility of the schedule is faster during the first 250 generations and then slows down considerably. In other words, as better schedules are obtained, improvements are harder to find. This in turn makes the progress slower. Moreover, the randomly sampled points around the parent of the current generation show very small differences in the utility values. This makes it difficult for the ES to maintain a steady increase in the schedule utility.

Figure 4b shows the percentage global utility generated by the $(2+1)$-ES and the GA. Performance of both methods is on a par with the stochastic search variants for the *INC* size distribution. The GA's performance on the *DEC* distribution is easily overpowered by a $(1+3)$-ES. Further, the GA does a maximal utilization of the allowed function evaluations of 15000. The $(2+1)$-ES achieves an utility of 83%, equivalent to that of the $(1 + 5)$-ES with 2000 generations. An important factor to consider here is the number of function evaluations used in the two methods – 5000 in $(2 + 1)$-ES compared to 10000 in the $(1 + 5)$-ES. Although this performance is marginally better (about 3%) than the $(1+5)$-ES with 1000 generations, we stress that even obtaining such marginal improvements is difficult with the *DEC* distribution.

The enhanced performance of $(2+1)$-ES is attributable to the additional exploration brought forth by the recombination operator. Local search methods do not display enough exploratory capabilities when stuck in a plateau of the search space. Recombination allows for a more diverse sampling in such cases, thereby resulting in a faster exploration through plateaus. The GA is expected to benefit from this as well. The cause of its poor performance is not well understood.

Scheduling time is an important factor to consider in any dynamic scheduler. However, it should be noted that the amount of time a scheduler has to generate a schedule need not be always fixed. In this study, the scheduler is triggered only when a new request arrives. The scheduler otherwise remains idle, including the time when a broadcast is ongoing. In a real scenario, it may be beneficial to trigger the scheduler more often – for example when a new broadcast starts – and allow for more exploration of the search space.

7 Conclusions

Pervasive computing applications often need to broadcast grouped data objects such that the elements in the group satisfy a user specified ordering constraint. In addition, objects not served within a specific window may end up having near zero utility. We introduce a method of utility accrual for grouped data objects and use it to evaluate the effectiveness of a schedule. We argue that evolution strategy is a viable methodology to maximize the utility of broadcasts given the run time constraints of the application. We investigate three different methods– a simple stochastic local search using $(1+\lambda)$-ES and $(1,\lambda)$-ES, a $(2+1)$-ES with a modified Syswerda recombination operator, and a genetic algorithm. Our experiments suggest that the generation of an optimal schedule when the most requested group has the highest number of data items is a difficult problem to solve, often requiring a longer duration of search. However, recombination based ES appears to be particularly effective. The $(2+1)$-ES with the proposed recombination operator demonstrates the potential to generate better schedules without engaging in too many function evaluations, thereby providing a fair trade-off between the run time and utility objectives of a scheduler.

The problem considered in this paper assumes that data items are unique across different data groups. However, there exists other problems in pervasive environments where the data groups can have common data items. In future, we plan to investigate if the scheduling strategy identified here works equally well in such problem domains.

Acknowledgment

This work was partially supported by the U.S. Air Force Office of Scientific Research under contracts FA9550-07-1-0042 and FA9550-07-1-0403. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies of the U.S. Air Force or other federal government agencies.

References

1. Ravindran, B., Jensen, E.D., Li, P.: On Recent Advances in Time/Utility Function Real-Time Scheduling and Resource Management. In: 8th IEEE ISORC, pp. 55–60 (2005)
2. Jensen, E., Locke, C., Tokuda, H.: A Time Driven Scheduling Model for Real-Time Operating Systems. In: 6th IEEE RTSS, pp. 112–122 (1985)
3. Buttazzo, G., Spuri, M., Sensini, F.: Value vs. Deadline Scheduling in Overload Conditions. In: 16th IEEE RTSS, pp. 90–99 (1995)
4. Wu, H., Balli, U., Ravindran, B., Jensen, E.D.: Utility Accrual Real-Time Scheduling Under Variable Cost Functions. In: 11th IEEE RTCSA, pp. 213–219 (2005)
5. Chehadeh, Y., Hurson, A., Kavehrad, M.: Object Organization on a Single Broadcast Channel in the Mobile Computing Environment. Multimedia Tools and Applications 9(1), 69–94 (1999)
6. Hurson, A., Chehadeh, Y., Hannan, J.: Object Organization on Parallel Broadcast Channels in a Global Information Sharing Environment. In: 19th IEEE IPCCC, pp. 347–353 (2000)
7. Huang, J.L., Chen, M.S.: Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment. IEEE Transactions on Knowledge and Data Engineering 16(9), 1143–1156 (2004)
8. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systemenach Prinzipien der biologischen Evolution. PhD thesis, Technical University of Berlin (1970)
9. Syswerda, G.: Schedule Optimization Using Genetic Algorithms. In: Davis, L. (ed.) The Genetic Algorithms Handbook (1990)
10. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web Caching and Zipf-Like Distributions: Evidence and Implications. In: IEEE INFOCOM, pp. 126–134 (1999)
11. Hameed, S., Vaidya, N.: Efficient Algorithms for Scheduling Data Broadcast. Wireless Networks 5(3), 183–193 (1999)
12. Lee, V.C., Wu, X., Ng, J.K.Y.: Scheduling Real-Time Requests in On-Demand Data Broadcast Environments. Real-Time Systems 34(2), 83–99 (2006)

A Multiobjective Evolutionary Algorithm for the Linear Shelf Space Allocation Problem

A.I. Esparcia-Alcázar[1], A.I. Martínez-García[1], J.M. Albarracín-Guillem[2], M.E. Palmer-Gato[2], J.J. Merelo[3], K.C. Sharman[1], and E. Alfaro-Cid[1]

[1] Instituto Tecnológico de Informática
Universidad Politécnica de Valencia, Spain
{anna,amartinez,ken,evalfaro}@iti.upv.es
[2] Departamento de Organización de Empresas
Universidad Politécnica de Valencia, Spain
{jmalbarr,marpalga}@doe.upv.es
[3] Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada, Spain
jmerelo@geneura.ugr.es

Abstract. This paper presents a multiobjetive approach to solve the Linear Shelf Space Allocation Problem (LiSSAP), which consists on allocating lengths of shelves in a given shop to specific products or groups of products. Previously we gave the first steps towards the development of a commercially viable tool that used evolutionary computation to address the problem; in this paper we introduce MELiSSA, standing for *M*ultiobjective *E*volutionary *Li*near *S*helf-*S*pace *A*llocation, and test it on two real problem configurations, yielding very good results.

1 Introduction

The problem of allocating space to a particular product in a shop is typically mapped in the literature to that of deciding which combination of products will yield the maximum profit. In general, the most commonly employed methods try and measure the impact on the customer of the relationship between allocated space and sales [1,2,3,8,9]. The aim is to find the allocation of space that maximises the profit. However, it is possible that this is just an academic pursuit and that in "real life" things are addressed in a different manner.

This is the case that we are going to address in this paper, which was presented to us by a well-known Spanish supermarket chain[1]. In this company the maximisation of the profit and the actual allocation of products to shelves were solved at different levels. At the management level the optimal (in terms of higher profit) lengths of shelves to

[1] We term this *Li*near *S*helf *S*pace *A*llocation *P*roblem, or LiSSAP, which differs from the Shelf-Space Allocation Problem (SSAP) described in [10] in that we aim to allocate products to *lengths of shelves* (horizontal allocation) and not to the individual tiers of the shelf (vertical allocation).

allocate to each product were determined. These were called the *category standards* and the set of category standards that configures a shop was termed the *standard shop*. At the shop level, the shop planners must take this "ideal" shop configuration (in practice a list of items and lengths of shelves) and transform it into an allocation of shelves to items, in effect adapting the standard shop to the actual space available. Because the size and layout of different premises can vary, the standard shop requirements cannot be fulfilled exactly, so for some categories the allocated shelf length must be increased or decreased with respect to the standard.

There are also other constraints for the placing of groups. Firstly, groups can have *affinities* or *adversities* between them. Furthermore, some groups must be placed near *reference points*, such as the oven, the freezer or the checkouts. Finally, groups must be placed together in a cohesive manner: it is preferable to avoid situations in which a group is scattered all over the shop, or with one or two modules isolated in the middle of a shelf.

The problem shop planners faced was the lack of suitable tools to perform this task, which meant that the allocation was done by hand. This resulted in several problems. Firstly, the allocation of products was likely to be suboptimal (in ways that we will define below). Secondly, no two shops had similar allocations, not only because the initial shop layout was different[2] but also because the allocation was performed by different persons, which could be very confusing for customers visiting different shops. Further, allocation by hand was time consuming (up to several days), which is unacceptable when many shops must be opened at the same time and also when things must be frequently rearranged due to the introduction of new products.

In previous work [4,5,7,11] we gave the first steps towards the development of such a tool by employing evolutionary algorithms. Here we take a step further by performing several methodological changes, the first of which regards the objective function: previously we were either using a single aggregative fitness function which encapsulated several objectives (giving different weight to each) [4,5,7] or addressed different constraints in sequence [11], via a separate algorithm; here we will employ a multiobjective approach and address all objectives simultaneously, which we consider more suited to the problem at hand.

The second change concerns the level of resolution of the problem, which in turn affects the representation (encoding). In our prior approach we assigned products or groups of products to modules (the parts in which shelves are divided). Here we will allocate groups to shelves and aisles; once this is done it is easy to allocate individual modules to groups, because in general there will be at most two groups per shelf.

The paper is laid out as follows: the variables that will be employed in the rest of the paper are described in Section 2. In Section 3 we describe the different aspects that are relevant to the MELiSSA methodology: encoding, fitness functions and operators. Results and analisys thereof are presented in Section 4. Finally, Section 5 summarises the conclusions and lays the lines for further work.

[2] We assume that the premises are rented and not built on purpose, which would result in a much easier allocation problem.

2 Modeling the LiSSAP

Given a set of G groups of products, a set of $\mathcal{A}$ aisles and a set of $\mathcal{S}$ shelves, with $\mathcal{S} \geq \mathcal{A}$. Each aisle a_i can contain one or two rows of shelves. Each shelf s_i, $i = 1 \ldots \mathcal{S}$, is part of one and only one aisle a_k, $k = 1 \ldots \mathcal{A}$, and row (of which there can be one or two), and is divided into M_i modules, with $M = \sum_{i=1}^{\mathcal{S}} M_i \geq G$ being the total number of modules in the shop.

Let PR be a number of reference points, which can be part of one or more aisles[3]. Let D be the $P \times \mathcal{A}$ matrix of distances between aisles, or *distance matrix*, whose components $d_{i,j}$ represent the distance between aisles a_i and a_j.

Let SS be the matrix that captures the standard shop requirements:

$$SS = \begin{pmatrix} std_1 & min_1 & max_1 \\ std_2 & min_2 & max_2 \\ \vdots & \vdots & \vdots \\ std_G & min_G & max_G \end{pmatrix} \tag{1}$$

where std_i, min_i and max_i denote the standard, minimum and maximum number of modules to allocate to the group g_i, respectively. We term *standard shop size*, S, the sum of the elements in the first column[4] of SS:

$$S = \sum_{i=1}^{G} std_i$$

3 Evolutionary Methodology for the LiSSAP

In this section we cover the main aspects of the algorithm: the chromosome encoding, the initialisation of the population, the fitness functions and the evolutionary operators.

Encoding. The individuals will consist of two chromosomes:

1. **Shelves chromosome.** It is represented as a vector of variable length rows, where each row corresponds to a group and the elements of the row are shelf identifiers, indicating the shelves occupied by the group.
2. **Modules chromosome.** As above, but in this case the elements of the row indicate the number of modules assigned to the group in the corresponding shelf

$$\mathcal{S}_{chrom} = \begin{pmatrix} S_{11} & \ldots & S_{1n} \\ S_{21} & \ldots & S_{2m} \\ \vdots & \vdots & \vdots \\ S_{G1} & \ldots & S_{Gk} \end{pmatrix}; \qquad M_{chrom} = \begin{pmatrix} M_{11} & \ldots & M_{1n} \\ M_{21} & \ldots & M_{2m} \\ \vdots & \vdots & \vdots \\ M_{G1} & \ldots & M_{Gk} \end{pmatrix} \tag{2}$$

[3] These will be modeled as fictitious modules to which we will allocate an equal number of fictitious groups.

[4] Note that, in general, the standard shop size S will differ from the actual shop size M, $S \neq M$.

Algorithm 1. Creation of an individual.

```
begin Algorithm create_individual
    [Assign initial values]
    [Assign reference points to reference groups]
    [Assign all modules of shelves to groups]
    for each shelf $s : 1 \dots \mathcal{S}$
        repeat
            Select a group $g$ at random
            Assign a feasible random number of modules to $g$
        until $s$ is fully occupied
    [Complete groups below $min$ from groups over $std$]
    for each group $g : 1 \dots \mathcal{G}$
        while($g$ is below its $min$) do
            Look for another group $g2$ above its $std$
            Transfer a feasible number of modules from $g2$ to $g$
end Algorithm;
```

Initialisation of the Population. Because we are only considering feasible individuals, the population initialisation method must ensure that the individuals created are valid. The pseudo-code for the initialisation is given in Algorithm 1.

Fitness Functions. To encapsulate the constraints described before we will define three fitness functions:

- ***Deviation fitness***, f_s, to capture the standard shop requirements. This function measures how close the allocation of number of modules to the groups is to the standard shop. The objective is to *minimise the absolute value of* f_s, which is given by

$$f_s = e^{-k}\,\frac{1}{G}\sum_{i=1}^{G} f_{s_i}$$

  where k equals the kurtosis of the deviations of all G groups[5] when this value is less than zero, and is zero otherwise and f_{s_i} is the deviation of group i. To obtain the value of f_{s_i} we will employ a function as shown in Figure 1.
- ***Affinity fitness***, f_a, which will represent the affinity requirements between groups. We have captured the affinity requirements in the following expression:

[5] We employ the kurtosis as a measure of how spread the distribution of deviations is. A high value of kurtosis indicates that all the deviations are around the same value, while a less than zero value indicates a flat distribution. Using this statistic we intend to avoid undesirable situations in which a few groups have a much higher deviation than the rest, as it is preferable that *all* groups have more or less the same values.

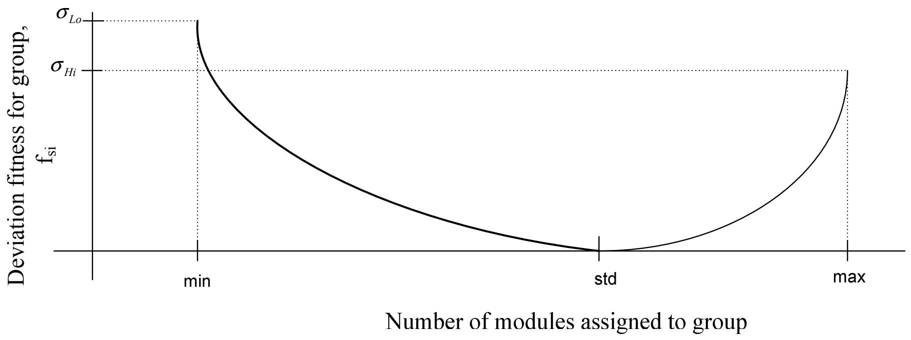


Fig. 1. Deviation function for a group g_i of given maximum, minimum and standard number of modules. The values of σ_{Lo} and σ_{Hi} are determined by experimentation. It must be pointed out that this function takes discrete values, in spite of having been represented as continuous.

$$f_a = \sum_{i=1}^{G} \sum_{j=1, j\neq i}^{G} \varphi \cdot f_{a_{i,j}}$$

where G is the total number of groups, $f_{a_{i,j}}$ is the affinity fitness between groups i and j and φ is a parameter determined by experimentation. The objective of the algorithm is to *maximise* f_a.

The affinity fitness between groups g_i and g_j, $f_{a_{i,j}}$, is defined as follows,

$$f_{a_{i,j}} = \begin{cases} \left(\frac{d_{i,j}}{d_{max}}\right)^{\alpha} & \text{if } g_i \text{ and } g_j \text{ are adverse} \\ 1 & \text{if } g_i \text{ and } g_j \text{ are indifferent} \\ \left(1 - \frac{d_{i,j}}{d_{max}}\right)^{\alpha} & \text{if } g_i \text{ and } g_j \text{ are affine} \\ \left(1 - \frac{d_{i,j}}{d_{max}}\right)^{\beta} & \text{if } g_i \text{ and } g_j \text{ are affine, with one being a reference point} \end{cases}$$

where the values of α and β have been determined experimentally, d_{max} is the maximum distance between aisles and $d_{i,j}$ is the distance between groups g_i and g_j.

– ***Dispersion fitness***, f_d, expressing the cohesion within a group. The dispersion fitness measures the spread of groups over the shelves and across the whole shop. The function we use for this purpose is :

$$f_d = \frac{1}{G} \sum_{i=1}^{G} f_{d_i}$$

where f_{d_i} is the dispersion for group i, calculated as follows:

$$f_{d_i} = \frac{1}{\mathcal{S}_i} \sum_{j=1}^{\mathcal{S}_i} \left(1 - \frac{n_{i,s_j} - 1}{n_i - 1}\right) \cdot \frac{\max d_{\mathcal{S}_i}}{d_{max}}$$

where $\mathcal{S}_i$ is the number of shelves occupied by group i, n_{i,s_j} is the number of modules of group i in shelf s_j, n_i is the total number of modules of group i and $\max d_{\mathcal{S}_i}$ is the maximum distance between any two shelves occupied by the group. The objective is to *minimise* f_d.

Table 1. Configuration of evolutionary algorithm

Number of generations	100
Size of population	500
Restart when	50% of individuals are repeated
Size of tournament	2
Maximum size of Pareto front	50
Number of individuals kept for final solution	5

These functions and the distance measures employed are fully described in [6]. It must be noted that the three fitnesses are not equally important for the allocation: the deviation fitness must take precedence because it is directly based on economic criteria.

Evolutionary Operators. These have been designed to allow only valid individuals. For simplicity reasons we have not defined a crossover operator and will only perform mutation. The following mutation operators were implemented:

- `give_a_shelf`: the group that occupies less modules in a shelf gives them to any other. The yielding group shall not remain under the minimum, nor the receiving above the maximum. This operator acts on both chromosomes.
- `swap_shelves`: two groups exchange all the modules each of them occupies, either in the same shelf or in two different shelves. This operator acts on both chromosomes.
- `give_modules`: a group gives another a random number of the modules it occupies in a shelf. This operator acts only on the M chromosome.

These operators are applied with different probabilities: `swap_shelves` will have a probability of 0.5 and the rest 0.25 each. If the mutation does not change the individual then another one is attempted up to a maximum of ten times.

Once the evolutionary run has finished, we have a Pareto front consisting of 50 non-dominated solutions. Because we cannot present the user with such a high number we must perform a selection. This will be based on the fact that the deviation fitness must take precedence over the other two, as stated above.

4 Experiments

The data set used to test the algorithm was taken from a well-known Spanish supermarket chain and consists of 8 groups, as shown in Table 2.

The difficulty of this problem lies in the fact that the biggest group (G2, *Cosmetics and household products*), i.e. the one that has a bigger standard value, is also adverse to most other groups. Hence, it is going to be hard for the algorithm to find a location for G2 that is far from nearly all others.

In order to test the algorithm two shelf layout configurations were chosen, which we have called Shop M154 and Shop M127RP. The former consists of 8 aisles, 17 shelves and $M = 154$ modules and the latter of 11 aisles, 19 shelves, and 127 modules. Hence, the shops are respectively greater and smaller that the standard shop size of 130 given

Table 2. Standard shop used in the experiments, with a size S of 130

Group #	Description	std	min	max	Affine to	Adverse to
1	Food - General	14	11	19	3,4,7	2,8
2	Cosmetics and household products	40	34	52	-	1,3,4,5,6,7
3	Food - Snacks	7	7	9	1	2,8
4	Sauces and seasoning products	9	7	11	1,7	2
5	Bakery, cereals	21	15	28	RP1	2,8
6	Baby food and accessories	13	8	19	-	2,8
7	Beverages	17	9	21	1,4	2,8
8	Pet food and accessories	9	5	11	-	1,3,5,6,7
Ref. point #	**Description**					
RP1	Oven (N.B. This will only be used for Shop M127RP)					

Table 3. A solution for Shop M154; `maximum` and `rangemax` indicate that the group is at its maximum number of modules or between the standard and the maximum number of modules respectively

G1: $S6(4)$	$S9(3)$	$S15(12)$			$\rightarrow$ 19	modules (`maximum`)
G2: $S4(4)$	$S7(12)$	$S8(12)$	$S13(12)$	$S14(12)$	$\rightarrow$ 52	modules (`maximum`)
G3: $S9(9)$					$\rightarrow$ 9	modules (`maximum`)
G4: $S1(10)$					$\rightarrow$ 10	modules (`rangemax`)
G5: $S3(4)$	$S10(12)$	$S11(12)$			$\rightarrow$ 28	modules (`maximum`)
G6: $S2(10)$	$S5(4)$	$S12(5)$			$\rightarrow$ 19	modules (`maximum`)
G7: $S12(7)$	$S16(12)$				$\rightarrow$ 19	modules (`rangemax`)
G8: $S0(10)$					$\rightarrow$ 10	modules (`rangemax`)

$$f_a = 35.60611,\ f_d = 0.31056547,\ f_s = 0.8333333$$

Table 4. A solution for Shop M127RP; `maximum`, `rangemax` and `rangemin` indicate that the group is at its maximum number of modules, between the standard and the maximum, or between the standard and minimum number of modules respectively

G1: $S1(8)$	$S5(5)$				$\rightarrow$ 13	modules (`rangemin`)
G2: $S0(8)$	$S3(8)$	$S13(8)$	$S14(16)$	$S15(6)$	$\rightarrow$ 36	modules (`rangemin`)
G3: $S11(8)$					$\rightarrow$ 8	modules (`rangemax`)
G4: $S2(8)$					$\rightarrow$ 8	modules (`rangemin`)
G5: $S4(8)$	$S10(8)$	$S16(6)$			$\rightarrow$ 22	modules (`rangemax`)
G6: $S7(5)$	$S8(5)$	$S9(5)$			$\rightarrow$ 15	modules (`rangemax`)
G7: $S12(8)$	$S17(6)$				$\rightarrow$ 14	modules (`rangemin`)
G8: $S6(5)$	$S18(6)$				$\rightarrow$ 11	modules (`maximum`)

$$f_a = 43.143826,\ f_d = 0.26538268,\ f_s = 0.0$$

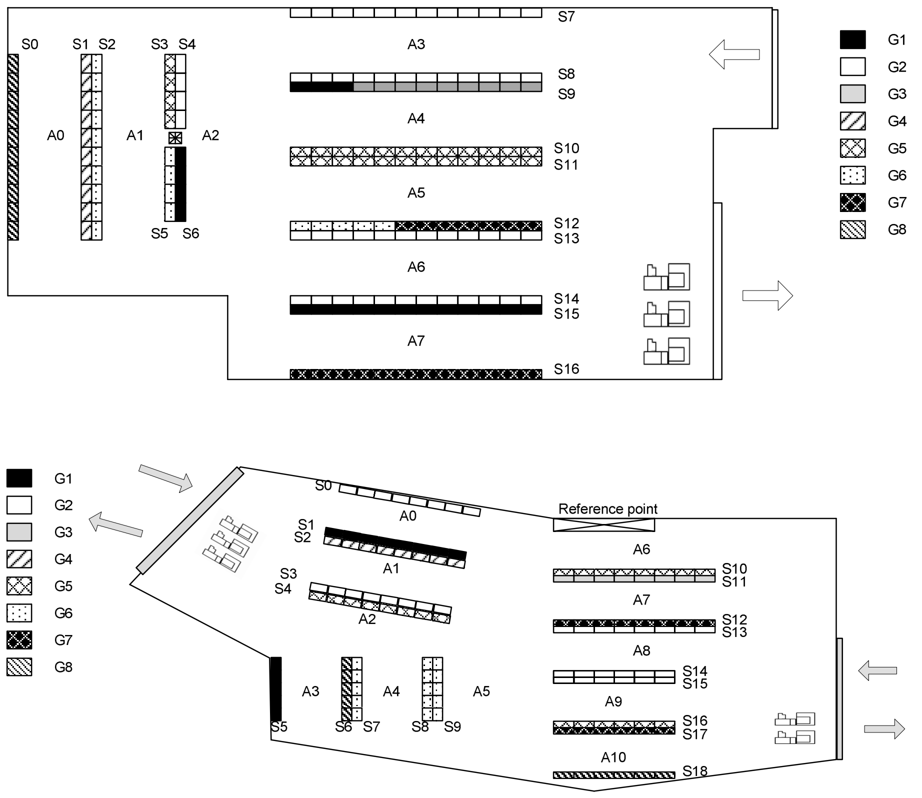


Fig. 2. Best solution obtained for Shop M154 (top) and Shop M127RP (bottom)

by Table 2. For Shop M127RP there is also one reference point, the oven, to which group G5 (*Bakery & cereals*) is affine. In both shops the module size is $1m \times 0.5m$ and the aisles between parallel shelves are 3 m wide.

We carried out 30 runs for each shop. The first thing that must be pointed out is that execution times are very short, less than half a minute per run for both shops.

The best solutions for each shop are given in Tables 3 and 4, where Si is the identifier of a shelf occupied by a given group and in brackets are the number of modules assigned to the group in that shelf; `maximum`, `rangemax` and `rangemin` indicate that the group is at its maximum number of modules, between the standard and the maximum, or between the standard and minimum number of modules respectively. For more solutions in the Pareto front the reader is referred to [6].

These two solutions (also depicted in Figure 2) have been selected following this rule: first the lowest deviation (in absolute value, i.e. the one closest to 0), then the highest affinity and finally the smallest dispersion. For Shop M154 this corresponds to run 4, whilst for Shop M127RP it corresponds to run 13. In the latter case, however, group G5 does not occupy the shelf closest to the reference point. For this reason we will choose instead the solution obtained in run 5, because its deviation is very similar to that of run 13 and its affinity and dispersion also have similar values. This illustrates the importance of user choice in the selection of the final solution.

For Shop M154, group G2 is placed on different aisles to all its adverse groups, except for shelf S4 which it is close to shelf S6, occupied by G1. On the other hand, affine groups are occupying close shelves. In this shop it can be seen that groups (except perhaps G5) are spread all over the shop.

In Shop M127RP the algorithm succeeds in placing G5 opposite the reference point. However, the nearest shelf occupied by it is S4, which is two aisles away. Here there is less dispersion than in Shop M154 because more groups are cohesive.

5 Conclusions and Future Work

We have introduced MELiSSA, a multiobjective evolutionary methodology to solve the LiSSAP, which incorporates a new model of the problem, a new encoding of the chromosome and three fitness functions. The latter in particular have proven to be extremely complex to tune and deserve further investigation.

We have observed that using the proposed method the groups tend to occupy complete shelves, as long as the size of shelves in the shop allows it. This is very important, since the undesirable situation in which a group is placed in an isolated module in a shelf never arises, which implies an improvement over the results presented in [11].

Regarding the affinities, the algorithm manages to avoid locating two adverse groups on the same shelf, although sometimes they can be placed on the same aisle. Further, group G5 (*Bakery & cereals*) tends to be located near or in front of the reference point.

If a shop size is above the standard, all groups will be at least in their standard and never below it. If a shop is smaller than the standard size, we cannot assure that the distribution of number of modules is uniform. It appears that using the kurtosis as a measure of the deviation spread works well when a shop is big enough and all groups can be over their standard values, because the algorithm will prefer a solution were all groups are in *std* at least. In the case of a shop size smaller than standard shop size, we cannot avoid that there is a group in its *max* whilst other groups are below their *std*.

In summary, it can be said that the proposed method is **effective** (in providing reasonable solutions) and **efficient** (by doing so in acceptable timescales).

Future work will involve greater refinements of the fitness functions, such as using the *centres of gravity* of each group to measure the distances in the affinity fitness, using a global dispersion function instead of considering separate dispersions for the groups and considering alternatives to the kurtosis as a measure of deviation.

Acknowledgement

This work was part of projects NABILA (supported by IMPIVA, Ref. IMCITA/2007/58, and partially funded by the European Regional Development Fund) and NoHNES (MEC, TIN2007-68083-C02).

References

1. Anderson, E.E., Amato, H.N.: A mathematical model for simultaneously determining the optimal brand collection and display area allocation. Operations Research 22, 13–21 (1974)
2. Bai, R., Kendall, G.: An investigation of automated planograms using a simulated annealing based hyper-heuristic. In: Proceedings of the 5th Metaheuristics International Conference (MIC 2003), pp. 25–28 (August 2003)
3. Corstjens, M., Doyle, P.: A model for optimizing retail space allocations. Management Science 27(7), 822–833 (1981)
4. Esparcia-Alcázar, A.I., Lluch-Revert, L., Albarracín-Guillem, J.M., Palmer-Gato, M.E., Sharman, K.: Towards an evolutionary tool for the allocation of supermarket shelf space. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Seattle, USA, vol. 2, pp. 1653–1660 (2006) ISBN:1-59593-184-4
5. Esparcia-Alcázar, A.I., Lluch-Revert, L., Sharman, K., Albarracín-Guillem, J.M., Palmer-Gato, M.E.: An evolutionary algorithm for the product to shelf allocation problem. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Vancouver, Canada, pp. 10966–10972 (2006) ISBN: 0-7803-9489-5
6. Esparcia-Alcázar, A.I., Martínez-García, A.I.: Linear Shelf Space Allocation using a Multi Objective Evolutionary Algorithm. Complex Adaptive Systems Group Technical Report ITI-SAC-027, Instituto Tecnológico de Informática, UPV, Spain (April 2008), `http://cas.iti.upv.es/reports/iti-sac-027.pdf`
7. Albarracín Guillem, J.M., Palmer Gato, M.E., Esparcia Alcázar, A.I., Babiloni, E.: Modelización mediante programación matemática del problema de ubicación de productos en estanterías. In: Actas del XXX Congreso Nacional de Estadística e Investigación Operativa y IV Jornadas de Estadística Pública, pp. 213–225 (September 2007)
8. Hansen, P., Heinsbroek, H.: Product selection and space allocation in supermarkets. Eur. J. Oper. Res. 3, 474–484 (1979)
9. Cunli, L., Yiu-ming, C., Yuping, W.: A bi-objective model for shelf space allocation using a hybrid genetic algorithm. In: Proceeedings of the International Joint Conference on Neural Networks (IJCNN 2007), pp. 2460–2465 (2007)
10. Lim, A., Rodrigues, B., Zhang, X.: Metaheuristics with local search techniques for retail shelf-space optimization. Management Science 50(1), 117–131 (2004)
11. Lluch-Revert, L., Esparcia-Alcázar, A.I., Albarracín-Guillem, J.M., Palmer-Gato, M.E.: A new two-stage approach to solve the linear shelf space allocation problem. In: Tan, K.C., Xu, J.X. (eds.) Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, Singapore, September 2007, pp. 4475–4482. IEEE, Los Alamitos (2007)

A Memetic Algorithm for the Delineation of Local Labour Markets

Francisco Flórez-Revuelta[1], José Manuel Casado-Díaz[2], Lucas Martínez-Bernabeu[1], and Raúl Gómez-Hernández[1]

[1] Research Unit on Industrial Computing and Computer Networks, University of Alicante, P.O. Box 99, E-03080, Alicante, Spain
florez@dtic.ua.es,Lucas.Martinez@ua.es,rgomez@dtic.ua.es
http://www.dtic.ua.es/i2rc

[2] Institute of International Economics, University of Alicante, P.O. Box 99, E-03080, Alicante, Spain
jmcasado@ua.es
http://iei.ua.es

Abstract. Given a territory composed of basic geographical units, the delineation of local labour market areas (LLMAs) can be seen as a problem in which those units are grouped subject to multiple constraints. In previous research, standard genetic algorithms were not able to find valid solutions, and a specific evolutionary algorithm was developed. The inclusion of multiple ad hoc operators allowed the algorithm to find better solutions than those of a widely-used greedy method. However, the percentage of invalid solutions was still very high. In this paper we improve that evolutionary algorithm through the inclusion of (i) a reparation process, that allows every invalid individual to fulfil the constraints and contribute to the evolution, and (ii) a hillclimbing optimisation procedure for each generated individual by means of an appropriate reassignment of some of its constituent units. We compare the results of both techniques against the previous results and a greedy method.

Keywords: memetic algorithm, evolutionary computation, regionalization, zone design, combinatorial optimization.

1 Introduction

Local labour market areas (LLMAs) are geographical entities defined to serve as a territorial framework to design, implement and monitor effective labour market policies and statistical operations at sub-national levels. The success of these key policies crucially depends on the adequacy of the LLMAs delineation. According to the code of good practices established by Eurostat [1] to guide the selection of a specific procedure, the resulting LLMAs geography must be conformed by disjoint areas exhaustively covering a given territory, characterised by a high degree of self-containment in terms of travel-to-work trips (i.e. most workers in a specific LLMA must live in that area and most of the LLMA's employed residents should also work locally), and relatively homogeneous in population size (exceeding a minimum size constraint, for instance). The problem is therefore the grouping of basic spatial units (BSU) -such as districts, municipalities or counties- into functional areas so that the proportion of workers that cross their boundaries in their travel to work is low, while the number of defined areas is maximized. This problem is analogous to a Graph Partitioning Problem (GPP) where the optimal number k of partitions is unknown and the requisite of size homogeneity is relaxed or removed, so it is expected to be at least as hard as the standard GPP (that is NP-hard). Thus, an exhaustive resolution of the problem is not possible.

One of the more widely and successfully used official procedures is that of Travel-to-Work Areas (TTWAs) in the UK (it is fully described in [2], and has been applied with minor changes in other countries: [3], [4], [5], and [6]). This regionalization method can be defined as a greedy algorithm that iteratively aggregates a given set of BSUs based on the relative attraction (in terms of commuting flows) between them until all the defined functional areas meet both self-containment and size constraints (in terms of employed population). The method allows reaching adecuate solutions with little CPU time.

In order to get solutions closer to the optimal, an evolutionary approach [7] was designed. The multiple constraints which are part of the problem cause the number of valid solutions –those that meet the constraints– to be extraordinarily small with regards to the search space, so standard genetic operators didn't lead to valid solutions in a reasonable lapse of time. This is the reason why an extensive set of specific crossover and mutation operators was proposed [8]. Whilst some of them have similarities with those used in other grouping and clustering problems, others are much more related to the very specific nature of the problem. However, and despite the design of *ad hoc* genetic operators, a high percentage of the individuals generated were not valid. In this paper we propose the inclusion of an intermediate stage in the evolutionary process aimed at 'repairing' every invalid individual. Additionally, we also include a stage of local optimization through optimal reassignment of BSUs chosen at random. The application of both techniques should accelerate the evolutionary process in terms of generations and help to reach solutions closer to the optimal one.

2 Problem Formulation

Let $S = \{S_1, S_2, \ldots, S_n\}$ be a set of BSUs (the territory to be divided into LLMAs) and W_{S_i,S_j} the number of commuters from BSU S_i to BSU S_j, that is, the number of residents in S_i that work in S_j (thus, W_{S_i,S_i} is the amount of people who simultaneously live and work whithin the boundaries of BSU S_i). The objective is to obtain the set of markets (LLMAs) $M = \{M_1, M_2, \ldots, M_m\}$, where m is unknown *a priori*, so as $M_i \neq \emptyset, \forall M_i \in M$; $\bigcup_{i=1}^{m} M_i = S$ and $M_i \cap M_j = \emptyset, \forall i,j \in [1,m], i \neq j$, $1 \leq m \leq n$), that maximizes fitness function f. Let II be the interaction index between two markets:

$$II(M_i, M_j) = \underbrace{\frac{W_{M_i,M_j}}{R_i}}_{PE_{M_i,M_j}} \times \underbrace{\frac{W_{M_i,M_j}}{J_j}}_{PJ_{M_i,M_j}} + \underbrace{\frac{W_{M_j,M_i}}{R_j}}_{PE_{M_j,M_i}} \times \underbrace{\frac{W_{M_j,M_i}}{J_i}}_{PJ_{M_j,M_i}} \quad (1)$$

where

$$W_{M_s,M_t} = \sum_{\forall S_i \in M_s} \sum_{\forall S_j \in M_t} W_{S_i,S_j} \tag{2}$$

is the total number of commuters residing in the set of BSUs of M_s that works in any of the BSUs of M_t; $R_k = W_{\{M_k\},S}$ the total number of workers residing in M_k; and $J_k = W_{S,\{M_k\}}$ the total number of jobs in M_k.

Factor PE_{M_i,M_j} is the fraction of the employed population residing in M_i and working in M_j; and PJ_{M_i,M_j} is the portion of jobs in M_j that are held by workers residing in (coming from) M_i.

This interaction index can be the base for different fitness functions. Among them in this exercise we have decided to test our method with

$$f(M) = card(M) \times \sum_{\forall S_i \in S} II(\{S_i\}, M_{S_i} - \{S_i\}) \tag{3}$$

where M_{S_i} is the market S_i belongs to. In our case we calculate the interaction index between a BSU S_k –which is considered as a mono-BSU market– and the market that would result if that BSU Sk is substracted from the market M_k it belongs to. This interaction index between a BSU and the market it belongs to is a generalization of the interaction index used in [2]. The inclusion of the number of LLMAs as a factor allows to reach the highest possible number of independent LLMAs –this is one of the criteria usually applied in practical exercises [9].

Besides, each market $M_i \in M$ must fulfil two requirements in terms of minimum self-containment percentages (β_1, β_2, $0 \le \beta_1 \le \beta_2 \le 1$) –i.e. both the proportion of the occupied working locally, and the proportion of jobs filled by local workers must exceed a given threshold–, and minimum size in terms of employed population (β_3, β_4, $1 \le \beta_4 \le \beta_3$):

$$min\left(\frac{W_{M_i,M_i}}{W_{M_i,S}}; \frac{W_{M_i,M_i}}{W_{S,M_i}}\right) \ge \beta_1 \tag{4}$$

$$W_{M_i,S} \ge \beta_4 \tag{5}$$

Very urbanized environments are in real world characterised by the intensity and complexity of the network of commuting flows, something which makes it difficult to identify isolated groups of BSU. To facilitate the identification of a larger number of separate LLMAs in such environments a trade-off between both constraints (self-containment and minimum size) has been introduced similarly to [2], but using the formulation proposed by Casado-Díaz[9]. According to this proposal, the minimum self-containment requirement is linearly relaxed from β_2 to β_1 for populations sizes from β_4 to β_3. For each market in a given solution, this trade-off is evaluated as follows:

$$\frac{min\left(\frac{W_{M_i,M_i}}{W_{M_i,S}}; \frac{W_{M_i,M_i}}{W_{S,M_i}}\right)}{\beta_2 - m\beta_4 + mW_{M_i,S}} \ge 1 \tag{6}$$

$$m = \frac{\beta_2 - \beta_1}{\beta_4 - \beta_3} \tag{7}$$

We have also included a minimum connectivity requisite to guarantee some degree of territorial contiguity without employing spatial data: a BSU can only belong to a market if it is reachable from any other BSU of that market through the γ largest outgoing/incoming commuting flows of each BSU in the market (we call this functional neighbourhood).

3 Evolutionary Proposal

The structure of the initial evolutionary algorithm for the regionalization of a given territory follows the next steps:

Step 1. Produce an initial population consisting of n_p individuals. At least one of the individuals in the population must be valid -i.e. it must meet all the constraints. To assure this the first individual generated consists of a single market covering all the territory. Complete the initial population with $n_p - 1$ randomly generated individuals (in practice, all of these are invalid solutions).

Step 2. Evaluate fitness of all individuals and sort them accordingly.

Step 3. Repeat n_r times: select two valid individuals from the current population by fitness-proportional probability, select with uniform probability one crossover operator, apply it to the two selected parents generating a new individual, and evaluate its fitness.

Step 4. Sort the whole population, composed of $n_p + n_r$ individuals, by their fitness value.

Step 5. Repeat n_m times: select a valid individual from the current population (including the new offspring from the recombination stage) by fitness-proportional probability, select with uniform probability one mutation operator, apply it to the selected individual generating a new individual, and evaluate its fitness.

Step 6. Sort the whole population, composed of $n_p + n_r + n_m$ individuals, by their fitness value.

Step 7. Select the population for the next generation choosing the n_p best individuals (truncation scheme).

Step 8. Stop condition: if the best individual has changed in the last g generations, return to step 3. Otherwise, finish.

3.1 Genetic Representation

The individuals which constitute the population represent feasible solutions, that is, the aggregation of all the BSUs composing territory S, into non over-lapping LLMAs. We have used a group-number encoding [10] where each individual is represented by a vector of n components, each of which corresponds to a BSU of S, and takes the value of the identifier of the market the BSU belongs to (Figure 1). This representations ensures the non-overlapping constraint is fulfilled.

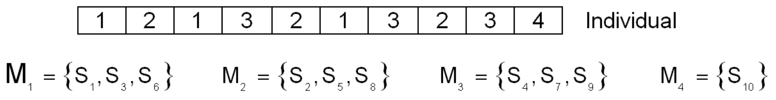

1 2 1 3 2 1 3 2 3 4
Individual
$M_1 = \{S_1, S_3, S_6\}$
$M_2 = \{S_2, S_5, S_8\}$
$M_3 = \{S_4, S_7, S_9\}$
$M_4 = \{S_{10}\}$

Fig. 1. Representation of individuals

3.2 Selection

The selection of the individuals to be affected by recombination and mutation operations is performed following a ranking method, according to which those individuals scoring higher in the fitness function have a larger probability of being selected. We use truncation for the selection of surviving individuals that compose the population in the next generation. So for every generation, the population is composed by the n_p better solutions in the previous generation.

3.3 Genetic Operators

Due to the large number of constraints that the individuals must meet, the usual operators of recombination and mutation seldom lead to valid solutions. This makes the evolution difficult or even unable to progress. For this reason we designed four *ad hoc* crossover operators and eleven mutation operators (see [8] for a detailed description). Specialized crossover operators consider the codification of both parents when the offspring is generated avoiding discrepancies between all of them. On the other hand, mutation operators have four main functions: division of markets, fusion of markets, reassignment of single BSUs, and reassignment of groups of BSUs. The goal of division operators is to increase the number of markets –i.e. $card(M)$– in the regionalization, so as to improve the detail of the result. Fusion operators merge markets to go back in the process of division. Reassignment operators try to improve the solution by reassigning specific BSUs between markets in a local search procedure.

3.4 Summary of the Evolutionary Process

To summarize, from an individual in which all the BSUs are merged to conform a single LLMA, successive applications of division and aggregation of markets, reassignment of single BSUs or groups of BSUs between markets, and recombinations, allow increasing the number of LLMAs, assigning the basic geographical units to the relevant LLMA so that the fitness function is maximised.

4 Memetic Proposal

One of the main problems of the application of this evolutionary algorithm, caused by restrictions that must be fulfilled, is the high percentage of invalid individuals resulting from the use of stochastical operators, where little or none information about the problem is used.

To achieve a higher success rate for those operators without complicating their algorithms, we implement a repair stage after each genetic operator. In this step invalid markets are disaggregated and their BSUs reassigned –among the rest of markets that form the individual being repaired–, until all the markets in the individual meet the constraints. As a side effect, this technique makes all randomly generated individuals in the initial population to be valid, and therefore the inclusion of an specific valid individual in step 1 to start the evolution is unnecessary.

Moreover, just after its evaluation, every valid individual goes on to a local search process, by means of the reassignment of individual BSUs among the existing markets, in order to optimize the generated individuals.

4.1 Repair of Invalid Individuals

The process of repair of an individual is based on the technique used by Coombes et al. [2] when a market does not fulfil the established restrictions. These invalid markets are successively disintegrated into their constituent BSUs which are then reassigned to the market with which they have the larger mutual interaction.

In an algorithmic way the process of repair would be as follows:

1. Verify the fulfilment of the constraints of each market $M_i \in M$ (see 3). If there is no invalid market, repair is finished.
2. Chose at random an invalid market M_i and disintegrate it.
3. Assign each BSU $S_k \in M_i$ (chosen in random order) to the market M' with which it possesses more interaction. That is:

$$M' = \underset{\forall M_j \in M, M_j \neq M_i}{\arg\max} \; II(\{S_k\}, M_j) \tag{8}$$

4. Once all the BSUs in M_i are reassigned, return to step 1.

As a result of this process, the individual will fulfil the constraints (unless there are not valid solutions in the search space due to excessive restriction), but the total number of LLMAs will decrease at least in one (if the individual was invalid). This worsens the fitness, since the specific function that we use in this exercise (3) tends to overvalue the solutions with larger number of markets (there is a preference for detail). Note that the final number of markets disaggregated in a repaired individual can be lower than the number of invalid markets before repair, because the reassigned BSUs of the first disintegrated markets can turn some other invalid markets into valid ones before the repair process choose them for disaggregation.

4.2 Improvement of Valid Individuals

We have also included a process of improvement of the generated individuals that fulfil all the restrictions or those that have been repaired. The goal is to carry out a local search to improve the fitness of the individual by reassigning single BSUs between markets. The process starts by randomly selecting a market

(LLMA) to be optimised. Next, the BSU belonging to that market and having the lowest interaction with the rest of its constituent BSUs, is reassigned to the market with which it has more interaction. This process continues while it increases the value of the fitness function. Likewise, we have also included a tolerance parameter that allows a number of unsuccessful reassignements before stopping the optimization process.

The process of optimization is as follows:

1. Set counter of unsuccessful attemps to $c = 0$.
2. A market $M_i \in M$ of the individual I is selected at random.
3. The BSU to be removed is selected as:

$$S_r = \underset{\forall S_j \in M_i}{\arg\min}\, II(\{S_j\}, M_i - \{S_j\}) \tag{9}$$

4. In a new individul I', copy of I, BSU S_r is assigned to its optimal market following equation (8).
5. If the new individual I' is invalid, it is repaired.
6. If $f(I') > f(I)$ (the new individual is better than the original), $I = I'$. Else increment the counter of failed attempts, $c = c + 1$
7. If $c < \xi$ return to step 2.

4.3 Results

To test our proposal we use a case study: the delineation of a set of LLMAs in the Region of Valencia, Spain. Travel-to-work data derived from the Spanish Census of Population [11] allowed us to build a 541 × 541 origin-destination commuting matrix (where 541 is the number n of municipalities that integrate the territory), where each cell represents W_{S_i,S_j}. Parameters were set in these values: size population $n_p = 100$, recombination offspring $n_r = 10$ and mutations $n_m = 24$. The condition of termination, i.e. generations without changes in the best individual is set to $g = 250$. Parameter γ of minimum flow connectivity or functional neightbourhood is set to 5. In the memetic algorithm, parameter ξ of allowed failed attemps in the optimization process is set to 5.

These evolutionary proposals substantially improve the results obtained by the traditional methods in both number of markets and fitness function (Table 1). The memetic algorithm obtains better mean results than the original evolutionary proposal and their dispersion is smaller. In an ANOVA test with confidence level 5%, we have obtained an F value equal to 16.04, greater than the critical value 3.86. So, the improvement of the memetic proposal is significant, although the best result was obtained with the original EA. However, although the solutions are obtained in less iterations, evolution time is around four times higher due to the great percentage of individuals that must be evaluated (Table 2) in the course of the reparation and optimization steps.

One of the consequences of the application of this processes of repair and optimization is that the success rate of the different operators –measured in terms of individuals generated by that individual that remain in the population at the

Table 1. Comparison of results

		Fitness value	Number of LLMAs	Generations	Time consumed (s.)
TTWAs method		120.23	44	-	1
Original EA [7]	**Best**	190.01	62	3620	1068
	Mean	180.54	59.52	3088.1	853.43
	σ	4.19	1.24	588.09	166.76
Memetic algorithm	**Best**	189.64	62	2580	4478
	Mean	182.04	59.43	2074.7	3332.58
	σ	3.21	1.14	566.84	894.44

Table 2. Percentage of individuals repaired and improved

	Percentage
Repaired by failing to fulfil contiguity constraint	53.44%
Repaired by failing to fulfil eq. (4) to (6)	35.50%
Improved individuals	7.58%

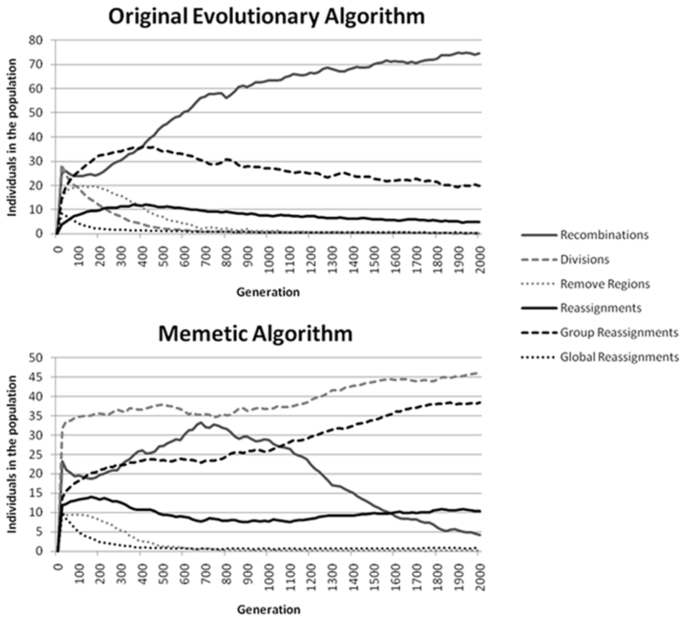


Fig. 2. Number of individuals in the population generated with the different operators

beggining of each generatio– differs from the original algorithm (Figure 2). For instance, in the evolutionary proposal division operators were successful in the beginning of the evolution. Once the number of markets reached its maximum, division of markets led to invalid individuals because they did not fulfil the size constraint. However, with the memetic approach these markets can be repaired and become successful.

5 Conclusions and Current Works

We have presented a memetic version of our previous evolutionary algorithm for delineation of functional areas (an unsupervised multi-constrained graph partitioning problem), and compared both results. Given the complexity of the problem, when the requirements associated to such a procedure are applied in real case studies (where the number of base spatial units is frequently very high), conventional genetic operators hardly ever lead to valid solutions. We tried to avoid this problem by designing ad-hoc operators. These specialized operators allowed to obtain good final delineations. However, the percentage of invalid generated individuals continued being very high. Our memetic extension of the EA includes a repair procedure to turn these solutions into valid ones that can contribute to the evolutionary process, and a local search optimization procedure. Both techniques result in a faster evolutionary process, in terms of generations. Besides, some of the operators that in previous versions were of low usefulness in the course of the evolution, are now much more significant in the whole process. The time consumed by the whole memetic approach is however comparatively very high. The repair and optimization procedures, and the numerous extra evaluations associated to them, makes the process longer. And opposite to our first estimations, the improvement in quality of the solutions is small, although statistically significant. This can be consequence of the high value that the objective function chosen in this exercise assigns to the number of delimited markets, since the repair process tends to reduce that value. Moreover the optimization process uses itself the repair process in its iterations (so it is time consuming) and it is based in one of the mutation operators already working in the mutation stage (so its effectiveness would be improved if that mutation operator is disabled). So now we are studying how this MA performs with other fitness functions and representations, and how we could optimize the repair and optimization processes.

We are also considering other ways of improvement, like the use of an adaptive scheme for the probability of application of the genetic operators in order to take advantage of the changing efficiency of the operators during the evolution, an island model for the parallel implementation of the algorithm, a reformulation of the problem based on multiobjective optimization, and the application of the method to other GPPs.

Acknowledgments. José M. Casado-Díaz has received financial support from the Spanish Ministry of Transports (Ministerio de Fomento) for the project "New methods for the analysis of the determinants and effects of commuting flows (MOVICO)" under Grant T 75/2006 (Spanish National Plan of A+D+i). Lucas Martínez-Bernabeu acknowledges financial support from the Spanish Ministry of Education and Science, the European Social Fund (ESF) and the University of Alicante. Raúl Gómez-Hernández acknowledges financial support from the University of Alicante.

References

1. Coombes, M.G.: EUROSTAT: Étude sur les zones d'emploi, Document E/LOC/20, Office for Official Publications of the European Communities, Luxembourg (1992)
2. Coombes, M.G., Green, A.E., Openshaw, S.: An efficient algorithm to generate official statistical reporting areas: the case of the 1984 Travel-to-Work Areas revision in Britain. Journal of the Operational Research Society 37, 943–953 (1986)
3. ISTAT, I sistemi locali del lavoro 2001 (2005), `http://www.istat.it`
4. Casado-Díaz, J.M.: Local Labour Market Areas in Spain: A Case Study. Regional Studies 34(9), 843–856 (2000)
5. Papps, K., Newell, J.O.: Identifying Functional Labour Market Areas in New Zealand: A Reconnaissance Study Using Travel-to-Work Data, Institute for the Study of Labor (IZA), Bonn, Discussion Paper n. 443 (2002)
6. Watts, M.: Local Labour Markets in New South Wales: Fact or Fiction?, Centre of Full Employment and Equity, The University of Newcastle, Australia, WP 04-12 (2004)
7. Flórez-Revuelta, F., Casado-Díaz, J.M., Martínez-Bernabeu, L.: An Evolutionary Approach to the Delineation of Functional Areas Based on Travel-to-work Flows. International Journal of Automation and Computing 05(1), 10–21 (2008)
8. Flórez-Revuelta, F., Casado-Díaz, J.M., Martínez-Bernabeu, L.: Specific crossover and mutation operators for a grouping problem based on interaction data in a regional science context. In: IEEE Congress on Evolutionary Computation, Singapore, pp. 378–385. IEEE, Los Alamitos (2007)
9. Casado-Díaz, J.M., Coombes, M.G.: The Delineation of 21st Century Local Labour Market Areas (LLMAs). In: Proceedings of the 8th Nectar Conference, Las Palmas de Gran Canaria, Spain (unpublished, 2005)
10. Jones, D.A., Beltramo, M.A.: Solving partitioning problems with genetic algorithms. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 442–449. Morgan Kaufmann Publishers, San Francisco (1991)
11. Instituto Nacional de Estadística. Censo de población (2001), `http://www.ine.es`

Evolving XSLT Stylesheets for Document Transformation[*]

P. Garcia-Sanchez, J.J. Merelo, J.L.J. Laredo, A.M. Mora, and P.A. Castillo

GeNeura Team, Dept. ATC, Universidad de Granada, Spain
{pgarcia,jmerelo,juanlu,amorag,pedro}@geneura.ugr.es

Abstract. This paper presents a new version of an evolutionary algorithm that creates XSLT programs from its intended input and output. XSLT is a general purpose, document-oriented functional language, generally used to transform XML documents (or, in general, solve any problem that can be coded as an XML document). Previously, a solution that solved the problem efficiently was proposed. In this paper, we improve on those results by testing different fitness functions, adding a new operator and changing the type of desired output document that can be obtained. The experiments show that the best results are obtained without considering the XSLT length and including this new operator.

1 Introduction

Since the Information Technology industry has settled on different Extensible Markup Language (XML) dialects as information exchange format, there is a business need for programs that transform from one XML set of tags to another, extracting information or combining it in many possible ways; a typical example of this transformation could be the extraction of news headlines from an on-line newspaper that uses XHTML.

XSLT stylesheets (XML Stylesheet Language for Transformations) [1], also called *logicsheets*, are programs designed for this purpose: applied to an XML document, they produce another. There are other possible solutions: programs written in any language that work with text as input and output (using, for instance, regular expressions) or SAX filters [2], that process each tag in a XML document in a different way, and do not need to load into memory the whole XML document. However, these solutions require programming in external languages, while XSLT is a part of the XML set of standards; in fact, XSLT logicsheets are XML documents. This is why XSLT is one of the most common ways of specifying document transformations. XSLT make use of XPath expressions [3] to select nodes from the source document.

The work needed for logicsheet creation scales quadratically with the number of input and output formats: for n input and m output formats, $n \times m$ transformations will be needed. Considering that each conversion is a hand-written program and the initial and final formats can vary with certain frequency, any

[*] Supported by projects TIN2007-68083-C02-01, P06-TIC-02025 and OTRI-1515.

automation of the process means a considerable saving of effort on the part of the programmers[1].

So, the problem is to find the XSLT logicsheet that, from one input XML document, is able to obtain an output XML document which contains exclusively the information desired from the first one. This information may be sorted in any possible way (possibly in an order different to the input document). In this work, an Evolutionary Algorithm (EA) [4] to resolve this problem is presented. The logicsheet will be evolved using evolutionary operators that will take into account the structure of the program and its components.

Thus, XSLT provides a general mechanism for the association of patterns in the source XML document to the application of format rules to these elements, but in order to simplify the search space for the evolutionary algorithm, only three instructions will be considered in this paper: template, which selects the XML fragments that will be included when the element in its match attribute is found; apply-templates, which is used to select the elements to which the transformation is going to be applied and delegate control to the corresponding templates; and copy-of, which includes the text representation of the nodes of the input XML into the output file (that is, copy all node contents and tags), so the output file will be a complete XML instead a list of content, as we did in our previous work. Of course, XPath expressions will also be used to select particular elements and sets of them.

In a previous work [5], we published an initial set of XSLT evolution experiments, testing different document structures and operators. In this paper we will try to improve on those results, by using XML output documents with tree structure, instead of plain text-only documents. This means that output documents are composed of several nodes, which makes it easier to compare them with each other. So, the output XML will be a complete XML document with a (possibly sorted in a different way) list of nodes present in the original document.

The rest of the paper is structured as follows: the state of the art is presented in Section 2. Section 3 describes the solution presented in this work, with the novel elements introduced. Experiments with the automatic generation of XSLT stylesheets for different examples are described in Section 4, and finally the conclusions and possible lines of future work are presented in Section 5.

2 State of the Art

To our knowledge, there are few works related to the application of genetic programming techniques to the automatic generation of XSLT logicsheets; one of them, by Scott Martens [6], presents a technique to find XSLT stylesheets that transform a XML file into HTML by using genetic programming. Martens works on simple XML documents and uses the UNIX diff function as the basis for its fitness function. He concludes that genetic programming is useful to obtain solutions to simple examples of the problem, but it needs unreasonable execution

[1] And money by whoever hires them.

times for complex examples and might not be a suitable method to solve this kind of problems.

Schmidt and Waltermann [7] approached the problem taking into account that XSLT is a functional language, and using functional language program generation techniques on it, in what they call *inductive synthesis*. First they create a non-recursive program, and then, by identifying recurrent parts, convert it into a recursive program; this is a generalization of the technique used to generate programs in other programming languages such as LISP [8], and used thoroughly since the eighties [9].

A few other authors have approached the general problem of generating XML document transformations knowing the original and target structure of the documents, as represented by its DTD (Document Type Definition): Leinonen et al. [10,11] have proposed semi-automatic generation of transformations for XML documents, but user input is needed to define the label association. There are also freeware programs that perform transformations on documents from a XSchema to another one. However, they must know both XSchemata in advance, and are not able to accomplish general transformations on well formed XML documents from examples.

In our previous work [5], we presented an evolutionary algorithm to obtain a XSLT that extracts information represented in a output XML from an input XML. Several XSLT structures and operators were presented and studied. The main inconvenient of that work is the output XML file is a list of text elements instead of XML nodes, which would be much more useful to perform real XML transformations. Additionally, the existing operators apparently led to situations where evolutionary changes were quite difficult, so a new operator is proposed. In this paper, the desired XML output document includes a set of nodes (text and tags) extracted from the input document, which can be additionally processed to change the required output tags.

3 Methodology

The EA described here evolves XSLT stylesheets, which are generated using a set of operators and evaluated using a fitness function that is related to the difference between generated XML and output XML associated to the example. The way the algorithm works is shown in Figure 1. The solution has been programmed using JEO [12], an evolutionary algorithm library developed at University of Granada as part of the DREAM project [13], which is available from `http://www.dr-ea-m.org`.

Since the search space of possible stylesheets is exceedingly large, language grammar must be considered in order to restrict it and avoid syntactically wrong stylesheet generation. Due to this, transformations are applied to a predetermined stylesheet structure which was selected among three different ones in previous work [5]. An example of this structure is shown in Figure 2. This type of structure is more constrained than other types; and search is thus easier, since less stylesheets are generated. Despite the constraints, mutation and crossover are much more disruptive, generating a rougher landscape than before.

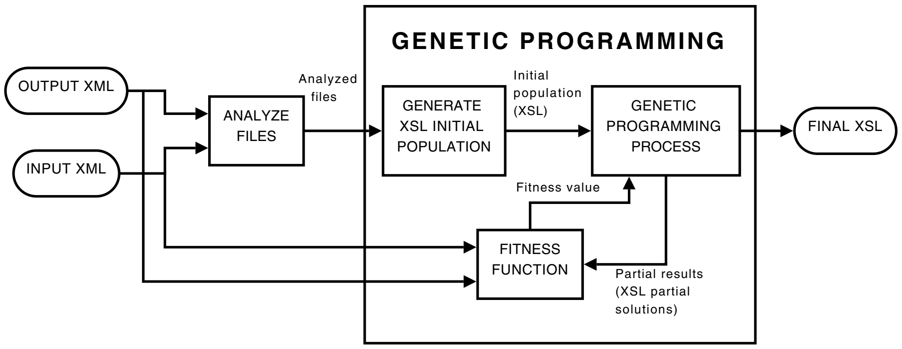


Fig. 1. This figure shows how the algorithm works. Each individual of the population is an XSLT stylesheet whose fitness is computed comparing the XML generated by the stylesheet (using the input XML) with the output XML.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output indent="no" method="xml"/>
  <xsl:template match="/">
    <grammar>
      <xsl:apply-templates select="/grammar"/>
    </grammar>
  </xsl:template>
  <xsl:template match="/grammar">
    <xsl:copy-of select="div[3]"/>
    <xsl:copy-of select="div[1]/h3[5]"/>
    <xsl:copy-of select="h1"/>
  </xsl:template>
</xsl:stylesheet>
```

Fig. 2. Example of a final XSLT generated by the algorithm. This logicsheet, applied to the input XML document, produces an XML document equals to the desired XML output document.

The operators may be classified in two different types: the first one consists in operators that modify XPath routes in the attributes of the XSLT instructions (`apply-template` and `copy-of`); and the other are the operators used to modify the XSLT tree structure. In order to ensure the existence of the elements (tags) added to the XPath expressions and XSLT instruction attributes, every time one of them is needed it is randomly selected from the input file. These operators have been described in more detail in our previous work [5], so we refer the interested reader to that paper. A complete list is shown in Table 1 whose names are quite descriptive. When an operator of the second group of the table is selected to modify an individual, another operator is selected randomly from the first group to use both in conjunction.

```
<xsl:template match="book">
    <xsl:copy-of select="chapter"/>
</xsl:template>
```

```
<xsl:template match="book">
  <xsl:copy-of select="chapter[1]"/>
  <xsl:copy-of select="chapter[2]"/>
  <xsl:copy-of select="chapter[3]"/>
  <xsl:copy-of select="chapter[4]"/>
</xsl:template>
```

Fig. 3. The left template is transformed into right template applying the split mutator. The number of chapters in the input XML were 4 (this is known by the algorithm when it process the input XML at the beginning of execution).

However, these operators are not enough to perform a smooth search; sometimes the XSLT search converges into a bad solutions when we want to select ordered but alternated items from a node. So it is necessary to add a new operator to increase the diversification of this solution. The new operator proposed, XSLTreeMutatorSplitTemplate, expands a random `copy-of` node into a list of complete `copy-of` with all cardinalities (as shown in Figure 3). The result of applying the new XSLT and the previous is the same, but it is easier for genetic operators to modify the list of copy-of than the generic one (modifying, adding or removing XPath and/or tags).

Since in this paper output is a fully formed XML document, fitness has been changed to be the XML difference between the desired and the obtained output, that is, the difference in nodes between the desired T and the actual document X. This difference breaks down in insertions (nodes in X but not in T) and deletions (nodes in T but not in X). We will leverage this vectorial structure of fitness so that evolution can profit from it: instead of using a single aggregative function, as we did in previous papers [5], fitness is now a vector that includes the number of node deletions and additions needed to obtain the target output from the obtained output, and the resulting XSLT stylesheet length. The XSLT stylesheet is correct only if the number of deletions and additions is 0; and minimizing length helps removing useless statements from it. So, fitness is minimized by comparing individuals as follows: An individual is considered better than another

– if the number of deletions is smaller,
– if the number of additions is smaller, being the number of deletions the same,
– if the length is smaller, being the number of deletions/additions the same.

Separating and prioritizing the number of deletions helps guide evolution, by trying to find first a stylesheet that includes all elements in the target document, then eliminating unneeded elements, while, at the same time, reducing length. However, this last element introduces selective pressure towards small stylesheets, which might hinder discovering the correct one, so we have also tested in this paper whether we should consider length or not as a part of the fitness.

4 Experiments and Results

To test the algorithm we have performed several experiments with 7 different XML input and output files. The algorithm has been executed 30 times for each input XML. Every experiment took 200.76 seconds in average to finish. The same input file was used for several experiments: a RSS feed from a weblog (`http://geneura.wordpress.com`) and an XHTML file. All input and output files and programs used in this experiment are available from our Subversion repository: `http://tinyurl.com/6nxv8c`.

Table 1. Operator priorities (used for the roulette wheel that randomly selects the operator to apply) used in the experiments

Operator	Priority
XSLTTreeMutatorXPathSetSelf	0.1
XSLTTreeMutatorXPathRemoveBranch	0.17
XSLTTreeMutatorXPathAddFilter	0.18
XSLTTreeMutatorXPathMutateFilter	0.18
XSLTTreeMutatorXPathRemoveFilter	0.2
XSLTTreeMutatorXPathAddBranch	0.16
XSLTTreeMutatorAddTemplate	0.2
XSLTTreeMutatorMutateTemplate	0.10
XSLTTreeMutatorRemoveTemplate	0.12
XSLTTreeAddApply	0.1
XSLTTreeMutateApply1	0.1
XSLTTreeMutateApply2	0.14
XSLTTreeRemoveApply	0.1
XSLTreeMutatorSplitTemplate	0.05
Probability of crossover	0.25
Probability of mutation	0.5

The computer used to perform the experiments is a Centrino Core Duo at 1.83 GHz, 2 GB RAM, and the Java Runtime Environment 1.6.0.01. The population size was 128 individuals for all runs, generated using the input XML as information source. The termination criteria was set to 300 generations or until a solution was found, and selection was performed via a 5-Tournament; 30 experiments were run, with different random seeds, for each input document. The XML and XSLT processors were the default ones included in the JRE standard library. The operator rates used in the experiments, which were tuned heuristically, are shown in Table 1. The crossover and mutation probability have been set to 0.25 and 0.5, after several experimental runs.

Due to the use of the new mutation operator we have performed the experiments using 3 different configurations. The first is the algorithm without the new operator (XSLTreeMutatorSplitTemplate), the second one, using the operator and the third without considering the length of the stylesheet in the fitness function. This helps to keep the solutions which have the same number of deletions and insertions but larger size caused by the use of the operator (expanding the copy-of tags). The breakdown of results per input file is shown in Table 2.

The fitness function, in general, yielded better results than previously. The algorithm was able to find an adequate XSLT stylesheet within the pre-assigned number of generations in most cases.

Table 2. Number of times, out of 30 experiments, a solution is found within the predefined number of generations without the split mutator, using the split mutator with normal fitness, and using split mutator without considering of the length of the generated stylesheet

Input file	Without Split	With Split	With Split w/o length
1	26	25	25
2	30	30	30
3	30	30	30
4	0	24	24
5	2	30	29
6	30	30	30
7	10	9	13

Table 3. Average generations/standard deviation to find an optimal solution (in less of 300 generations), without the split mutator, using the split mutator with normal fitness, and using split mutator without considering of the length of the generated stylesheet

Input file	W/o Split	With Split	With Split w/o length
1	62.86 ± 102.03	83.33 ± 112.81	66.33 ± 106.85
2	1.5 ± 1.25	1.6 ± 1.30	1.1 ± 1.29
3	4.13 ± 2.06	3.13 ± 1.94	3.83 ± 2.90
4	-	81.03 ± 112.25	71.68 ± 107.24
5	289.9 ±45.09	26.83 ± 5.35	36.03 ± 50.19
6	15.43 ± 5.74	20.43 ± 9.88	19.0 ± 11.12
7	232.17 ±105.48	245.46 ± 96.92	214.0 ± 112.90

Examples 1, 2, 3 and 6 are complete and ordered lists of elements of one or several nodes, whose solution is simple, since the algorithm can easily create a logicsheet that extracts all the childrens of a specific node. Example 7 takes specific and repeated elements from distinct nodes, which makes it more difficult, because different expressions are needed to extract each one of them and the generated logicsheet is more complex. Finally, examples 4 and 5 focus into portions of ordered and unordered fragments of an XML section (namely the 3rd and 6th chapters of a book), so the population converges into solutions with all the elements of a node (selecting all chapters of a book) due to the way the fitness works. Obtaining solutions for these examples is quite difficult without using the new operator (just two times for example 5 and none for 4), which is fixed when we use it; instead of selecting all chapters of a book (`book/chapter`), it selects

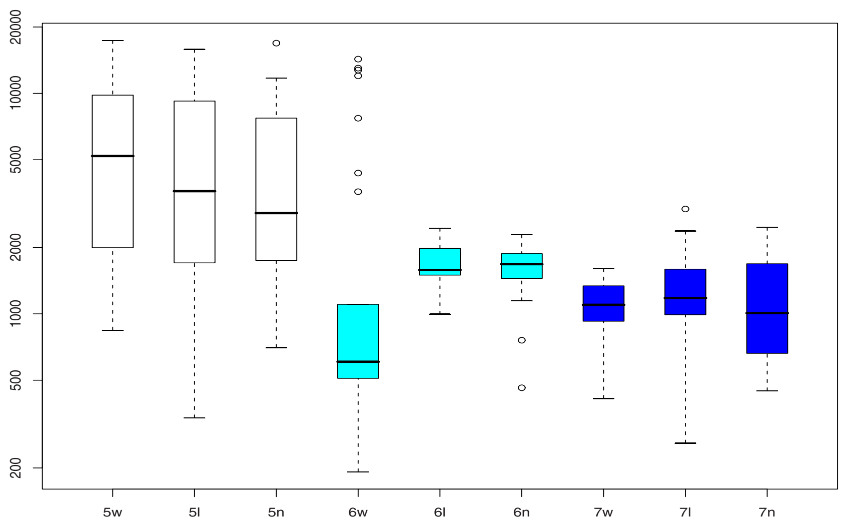


Fig. 4. Logarithmic boxplot of the number of evaluations to find the best individual in examples #5, #6 and #7 without split mutator (w), with split mutator considering length (l) and without this feature (n)

all chapters using XPath location (`book/chapter[1], book/chapter[2]...`), so it is easier for the algorithm to evolve this partial solution into a better one.

On the other hand, overruling the XSLT length comparison in fitness gives different solutions the same chances to evolve, so the algorithm maintains more diversity and finds solutions in less time than the cases comparing that length (see Table 3). However, the generated XSLT may contain useless statements, that could produce incorrect XMLs in a a production environment.

When a solution was found, the number of generations and time used to find it also varies, as shown in Table 3. In general, the exploration/exploitation balance seems to be biased towards exploration. Being such a vast and rough search space makes that, after a few initial generations that create stylesheets with a small difference from the target, mutations are the main operator at work.

5 Conclusions, Discussion, and Future Work

In this paper we present the results of an evolutionary algorithm designed to search the XSLT logicsheets that is able to make a particular transformation from an input XML document into a desired output one; one of the advantages of this application is that resulting logicsheets can be used directly in a production environment, without the interaction of a human operator. It tackles a real-world problem found in many organizations and it is open source software, available from `http://tinyurl.com/5lwjcn`.

The experiments have shown that the search space is particularly rough, with mutations in general leading to huge changes in fitness. The hierarchical fitness used is probably the cause of having a big loss of diversity at the beginning of the evolutionary search, leading to the need of a higher level of explorations later during the algorithm run. This problem will have to be approached via explicit diversity-preservation mechanisms, or by using a multiobjective evolutionary algorithm, instead of the one used now. A deeper understanding of how different operator rates affect the result will also help; for the time being, operator rate tuning has been very shallow, and geared towards obtaining the result. In addition, results showed in this paper can be used as a baseline for future versions of the algorithm, or other algorithms for the same problem. At any rate, unlike what was mentioned in the pioneering paper [6], solutions can be found effectively and efficiently.

However, there are some questions and issues that will have to be addressed in future papers:

– Using the DTD (associated to a XML file) as a source of information for conversions between XML documents and for restrictions of the possible variations.
– Adding different labels in the XSLT to allow the building of different kinds of documents such as HTML or WML.
– Testing evolution with other kind of tools, such as a chain of SAX filters.
– Obviously, testing different kinds and increasingly complex set of documents, and using several input and desired output documents at the same time, to test the generalization capability of the procedure.
– Using the identity transform [14] as another frame for evolution, as an alternative to the structure shown here. The identity transform puts every element found in the input document in the output document; elements can then be selectively eliminated via the addition of single statements.
– Tackle difficult problems from the point of view of a human operator. In general, the XSLT stylesheets found here could have been programmed by a knowledgeable person in around an hour, but in some cases, input/output mapping would not be so obvious at first sight. This will mean, in general, increase also the XSLT statements used in the stylesheet, and also in general, adding new types of operators.

References

1. Clark, J.: XSL transformations (XSLT), version 1.0, W3C recommendation November 16, 1999 (1999), `http://www.w3.org/TR/xslt.html`
2. Wikipedia: Simple API for XML — Wikipedia, the free encyclopedia [Online; accessed 21-March-2007] (2007)
3. Clark, J., DeRose, S., et al.: XML Path Language (XPath) Version 1.0. W3C Recommendation 16 (1999)
4. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Boston (1989)
5. Garcia-Sanchez, P., Laredo, J.L.J., Sevilla, J.P., Castillo, P., Merelo, J.J.: Improved evolutionary generation of XSLT stylesheets. ArXiV database (1999), `http://arxiv.org/abs/0803.1926`

6. Martens, S.: Automatic creation of XML document conversion scripts by genetic programming. In: Genetic Algorithms and Genetic Programming at Stanford, p. 269 (2000)
7. Schmid, U., Waltermann, J.: Automatic synthesis of XSL-transformations from example documents. In: Hamza, M. (ed.) IASTED International Conference on Artificial Intelligence and Applications, pp. 252–257 (2004)
8. Biermann, A.: The inference of regular LISP programs from examples. IEEE Transactions on Systems, Man and Cybernetics 8(8), 585–600 (1978)
9. Biermann, A.W., Guiho, G. (eds.): Computer Program Synthesis Methodologies, Reidel, Dordrecht (1983)
10. Leinonen, P.: Automating XML document structure transformations. In: Proceedings of the 2003 ACM Symposium on Document Engineering, pp. 26–28 (2003)
11. Kuikka, E., Leinonen, P., Penttonen, M.: Towards automating of document structure transformations. In: Proceedings of the 2002 ACM Symposium on Document Engineering, pp. 103–110 (2002)
12. Arenas, M.G., Dolin, B., Merelo-Guervós, J.J., Castillo, P.A., de Viana, I.F., Schoenauer, M.: JEO: Java Evolving Objects. In: Proceedings of the Genetic and Evolutionary Computation Conference, p. 991 (2002)
13. Arenas, M., Collet, P., Eiben, A., Jelasity, M., Merelo, J.J., Paechter, B., Preuß, M., Schoenauer, M.: A framework for distributed evolutionary algorithms. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 665–675. Springer, Heidelberg (2002)
14. Wikipedia: Identity transform — Wikipedia, The Free Encyclopedia (2007) [Online; accessed 24-January-2008], `http://en.wikipedia.org/wiki/Identity_transform`

Fast Multi-objective Scheduling of Jobs to Constrained Resources Using a Hybrid Evolutionary Algorithm

Wilfried Jakob, Alexander Quinte, Karl-Uwe Stucky, and Wolfgang Süß

Forschungszentrum Karlsruhe GmbH
Institute for Applied Computer Science
P.O. Box 3640, 76021 Karlsruhe, Germany
{wilfried.jakob,alexander.quinte,uwe.stucky,
wolfgang.suess}@iai.fzk.de

Abstract. The problem tackled here combines three properties of scheduling tasks, each of which makes the basic task more challenging: job scheduling with precedence rules, co-allocation of restricted resources of different performances and costs, and a multi-objective fitness function. As the algorithm must come up with results within a few minutes runtime, EA techniques must be tuned to this limitation. The paper describes how this was achieved and compares the results with a common scheduling algorithm, the Giffler-Thompson procedure.

1 Introduction

The problem is motivated by, but not limited to the task of scheduling jobs to the resources of a computational grid [1] in such a way that the partially conflicting interests of resource users and providers are satisfied as well as possible. Due to the dynamic nature of the grid, this is a permanent process and the time available for the scheduling is limited to a few minutes.

The scheduling task can be briefly characterised as follows: users describe their *application jobs*, consisting of one or more atomic *grid jobs*, by workflows, each of which may be regarded a directed acyclic graph (DAG) defining precedence rules between the grid jobs. They state what a grid job should do by requesting resources like software, storage capacity, and/or additional equipment. These resources may need other resources. For example, a software tool may require a certain operating system and appropriate computer hardware to run on. This leads to the concept of primary and dependent resources, the latter being requested by other resources rather than by grid jobs directly. The availability of all these resources is limited in a heterogeneous way, e.g. the amount of software licences will usually differ from the number of computers the software can run on. The availability of resources may be restricted to certain time periods per day or week according to the needs of the resource provider. The costs may also vary over time. Furthermore, the performance of the resources will usually differ, thus resulting in different cost-performance ratios.

Nearly all job scheduling tasks and algorithms deal with a single objective to be optimised like e.g. the makespan [2, 3]. However, this is not sufficient to fulfil the different needs of resource users and providers. Therefore, the following four objectives are considered: *completion time* and *costs* of each application job measured as fulfilment of user-given limits and averaged, and to meet the demands of resource providers, the *total makespan* of all application jobs and the ratio of *resource utilisation*. As the scheduling process is an automated procedure, a single solution is required instead of a set of alternative ones. Hence, the weighted sum is used with weights based on earlier experience. It also allows for an easy steering of the solution.

Generalising, this scheduling task requires co-allocation of heterogeneous resources of different performances, varying abilities (software), and with time dependant availability and costs. Precedence rules of the elementary jobs must be adhered to and time and cost limits should be fulfilled as well as possible. This task has some similarities to the *resource-constrained project scheduling problem* (RCPSP), one of the most ambitious tasks of operations research [4]. The RCPSP consists of a comparable scheduling task of jobs with precedence rules and alternative resources, but is subject to capacity constraints of the resources. The task on hand is of greater complexity as the RCPSP considers only one objective, the makespan, and as in the RCPSP the allocation time required by a particular job is equal for all resources.

As our task includes the job shop problem, it is NP-complete. Thus, approximated solutions can be expected only. First solutions are generated by some simple heuristics and the well-known Giffler-Thompson algorithm (**GTA**) [5]. It is the aim of our work to improve these solutions to the largest extent possible using a hybrid Evolutionary Algorithm (EA). The application of EA or other meta heuristics to scheduling tasks is nothing new, see e.g. [4], but the combination of scheduling multi-DAGs, constrained heterogeneous resources, multi-objective optimisation, and a short runtime makes it more or less special and unique, as pointed out in Section 2. This section also gives a formal definition of the problem and an overlook of related work. Section 3 describes the heuristics used, the necessary extensions of the GTA, and the EA together with the used gene models, repair mechanisms, and specialised crossover operators. These algorithms and models are compared in the experiments reported in Section 4. Section 5 contains a conclusion and an outlook.

2 Problem Definition and Related Work

We use a notation common to the scheduling literature [2, 4] to ease comparisons to other scheduling problems. Given are a set $M=\{M_1, \ldots, M_m\}$ of resources, a set $J=\{J_1, \ldots, J_l\}$ of application jobs, and a set O of grid jobs. The n GridJobs of application job J_i are denoted by $O_{i1}, \ldots, O_{in}$.. The following functions are given:

- a precedence function $p{:}O \times O \rightarrow \{TRUE, FALSE\}$ for the grid jobs
- an assignment function $\mu{:}O \rightarrow \mathcal{P}(\mathcal{P}(M))$ from grid jobs to resource sets. $\mathcal{P}(M)$ is the power set of M. μ_{ij} is the set of all possible combinations of resources from M, which together are able to perform the grid job O_{ij}
- a function $t{:}O \times \mathcal{P}(M) \rightarrow \Re$, which gives for every grid job O_{ij} the time needed for the processing on a resource set $R_{ij} \in \mu_{ij}$
- a cost function, $c{:}\Re \times \mathcal{P}(M) \rightarrow \Re$, which gives for every time z the costs per time unit of the given resource set

Optimisation is done by choosing suitable start times $s(O_{ij}) \in \Re$ and resource allocations $R_{ij} \in \mu_{ij}$. A solution is valid, if the following two restrictions are met:

1. All grid jobs are planned and resources are allocated exclusively:

$$\forall O_{ij} : \exists s(O_{ij}) \in \Re, R_{ij} \in \mu_{ij} : \forall M_j \in R_{ij} :$$
$$M_j \text{ is in } [\, s(O_{ij}); s(O_{ij}) + t(O_{ij}, R_{ij}) \,] \text{ exclusively allocated by } O_{ij}. \tag{1}$$

2. Precedence relations are adhered to:

$$\forall i, j \neq k : p(O_{ij}, O_{ik}) \Rightarrow s(O_{ik}) \geq s(O_{ij}) + t(O_{ij}, R_{ij}) \tag{2}$$

A violation of the two following constraints is treated by penalty functions in such a way that the amount of time and cost overruns is considered as well as the number of affected application jobs.

1. All application jobs J_i have due dates d_i, which must be adhered to:

$$\forall J_i : d_i \geq s(O_{in}) + t(O_{in}, R_{in}) \quad \text{where } O_{in} \text{ is the last grid job of } J_i \tag{3}$$

2. All application jobs J_i have a cost limit c_i, which must be observed:

$$\forall J_i : c_i \geq \sum_{j=1}^{n} \int_{s(O_{ij})}^{s(O_{ij})+t(O_{ij},R_{ij})} c(s, R_{ij})\, ds \tag{4}$$

The fitness is calculated as the weighted sum of the already mentioned four main objectives and one auxiliary objective, which is described here very briefly only due to the lack of space. It measures the average delay of each non-terminal grid job (i.e. a grid job with no successors) relative to the earliest starting time of its application job and it is aimed at rewarding the earlier completion of non-terminal grid jobs. The idea is to support the process of starting grid jobs earlier such that the final grid job can be completed earlier in the end which is recognised by the main objective *completion time.* We name the resulting fitness sum *raw fitness*, as it can be lowered by the application of one ore more penalty functions, each of which delivers a factor between 0 and 1, by which the raw fitness is multiplied to obtain the *end fitness*.

A comparable problem could not be found in literature, see e.g. [2] and [4] for a comprehensive presentation of scheduling problems. This corresponds to the results of the literature review found in [3]. There, it is concluded that only few publications deal with multiple objectives in scheduling and, if so, they mostly deal with single machine problems and Pareto optimisation. Of course, a lot of literature focuses on partial aspects of this problem. We will come back to some articles when already existing techniques incorporated in the used EA are presented.

3 Basic Algorithms, Two Gene Models, and Some EA Extensions

The original Giffler-Thompson algorithm [5, 6] is aimed at achieving a minimal makespan and a good rate of utilisation. It was extended for the problem on hand, such that it can deal with alternative resources, resource-dependant execution times, and multiple successors of one grid job. Costs are not taken into account by the original GTA and the used implementation does it only when the next set of resources is constructed and alternatives are available. We used some recommended [6] and some problem-specific priority rules, which are explained later with the results.

In addition to the GTA, the following heuristics are used to generate schedules. In a first step a sequence of grid jobs is produced by these three heuristic rules:

1. *Shortest due date*: grid jobs of the application job with the shortest due date first
2. *Shortest working time of grid job*: grid jobs with the shortest working time first
3. *Shortest working time of application job*: grid jobs of the application job with the shortest working time first

In the next step resources are allocated to the grid jobs using one of the following three resource allocation strategies (**RAS**):

RAS-1: Use the fastest resource of the earliest available for all grid jobs
RAS-2: Use the cheapest resource of the earliest available for all grid jobs
RAS-3: Use RAS-1 or RAS-2 for all grid jobs of an application job according to its time/cost preference

Processing of the three grid job sequences with these RAS, which can be computed very fast, yields up to nine different schedules.

Our *Global Optimising Resource Broker and Allocator* [7] performs a two-step planning process. In the first step the set of heuristics, including the GTA, is applied and the results are used to seed the start population of the subsequent EA run. As EA we use GLEAM [8], which already contains some evolutionary operators designed for combinatorial problems. Due to the lack of space, they are summarised only and the interested reader is referred to [8]. Apart from the standard mutation, which changes the sequence of genes by simply shifting one of them, GLEAM contains the movement of gene segments and the inversion of their internal order. A chromosome consists of a sequence of segments, containing a sequence of genes. As segment boundaries can be changed by some mutations, the segments form an evolvable meta structure over the chromosomes. Segment boundaries are also used for the 1- and n-point crossover operators, which include a genetic repair that insures that every offspring does not lack genes in the end. The evolvable segmentation and its associated operators among others distinguish GLEAM from most standard EAs.

Two gene models are compared. Both use one gene per grid job, which contains the grid job index at the minimum. The interpretation of a chromosome is done by processing the genes in the order of their appearance as described below.

Gene model **GM1** uses gene types with as many integer parameters as resources have to be co-allocated. The range of each parameter reflects the number of alternatively usable resources. A schedule is constructed from a chromosome as follows:

```
1. IF grid job has predecessors THEN
      determine the latest end time of all preceding grid jobs in startTime
   ELSE
      startTime equals to the earliest starting time of the application job
2. Calculate the duration of the job on the given resources.
3. Search for a free time interval of the calculated duration beginning at startTime on all
   selected resources and allocate them.
```

Gene model **GM2** is aimed at a reasonable reduction of the search space. It replaces the evolutionary selection of resources by the evolutionary selection of one of the RAS heuristics. This is done by an additional RAS gene and the crossover operators are modified to pass on the RAS gene of the better parent.

The genes of gene model **GM2** have no parameters but there is a new RAS gene instead. Its parameter defines the RAS to be used. Steps 2 and 3 of GM1 are now as follows:

2. Produce a list of alternatives for every primary resource according to the actual RAS.
3. Calculate the duration of the job beginning with the first resources of the lists and search for a free time slot for all first resources, including depending ones, beginning at *startTime*. If no feasible slot is found, the resources at the next position of the lists are used.
4. Allocate the found resources to the grid job with the calculated time frame.

The interpretation of the chromosomes described ensures that the precedence relations of grid jobs are not violated as long as no gene is located before the gene of its preceding grid job. As this may be interfered with by some genetic operators, two alternative repair mechanisms are applied and compared:

The *genetic repair* searches for all genes of grid jobs, the genes of the preceding grid jobs of which are not located on the chromosome before. Such a gene is shifted until all genes of preceding grid jobs are on prior positions. As a result, the mechanism may hamper meaningful steps of shifting genes. This is the explanation of the outcome of experiments earlier than those reported here, which produced best results when a fraction of about 20% of the offspring is corrected only. Using genetic repair therefore requires the application of an appropriate penalty function.

Phenotypic repair is aimed at a correct interpretation of a chromosome rather than altering it. If the processing of a gene tries to schedule a grid job with missing, already scheduled predecessors, it simply suspends the scheduling until all predecessors will have been scheduled. The advantage of this approach is that there are no faulty schedules and that intermediate steps of shifting genes, which itself may be faulty, are now allowed to occur and hopefully result in a better schedule.

Furthermore, in the experiment section we report about the effect of two crossover operators from literature, which are aimed at passing on sequence information. The well-known *order-based crossover* **OX** [9] preserves the relative order of the parent genes, while the *precedence-preserving operator* **PPX** [10] does this more strictly by perpetuating the absolute order. In contrast to OX, PPX ensures that sequence-correct parents produce sequence-correct offspring at the price of limited gene mixing.

4 Experiments

For the experiments, a set of benchmarks has been developed and they are evaluated using a simulated grid environment. This paper presents the results obtained with the standard benchmark set described in [7]. It consists of four classes of application jobs, each class representing different values of the following two characteristics: the degree of dependencies between grid jobs *D* and the degree of freedom of resource selection *R*. The four basic benchmark classes are abbreviated by *sRsD*, *sRlD*, *lRsD*, and *lRlD*, where *s* stands for small and *l* for large values of *R* and *D*. As the amount of grid jobs is another measure of complexity, benchmarks containing 50, 100, and 200 grid jobs were defined using the same set of resources for every class. A fourth benchmark set again consists of 200 grid jobs, but with a doubled set of resources available (abbreviated by *200d* in the figures).

Besides the structure and amount of jobs and resources, the tightness of time and cost restrictions is another essential characteristic of the complexity and complicacy of a scheduling task. These two limits are intentionally set in such a way that cost and time overruns (cf. eqs. (3) and (4)) are provoked for the heuristic procedures running before the EA, i.e. they are always invoking the corresponding penalty functions. Thus, a measure of success is defined for both the GTA and the EA: can schedules be found that adhere to the given restrictions and, if so, always or to what fraction of the EA runs? This fraction is denoted as *success rate* and represents the first and crucial measure of the experiments. The second is the quality of the schedules measured as the *fitness improvement* obtained from an EA run. For a fair judgment, the non-penalised raw fitness of heuristic planning is compared to the end fitness of the EA. The fitness values are also used to decide on the significance of differences observed. For judging the differences between heuristic and EA results, we check whether the best heuristic raw fitness is outside of the confidence interval (99% confidence) of the EA results. To assess different EA runs, the t-test is used. If it cannot be applied because of too large variance differences, the results are considered different.

The experiments are based on a runtime limit of three minutes (on one processor of an AMD Athlon 64 4400+ 2.0 GHz CPU), because this is considered a reasonable time frame for planning. It must be stressed that the complete investigation is based on the goal of achieving the best possible results within this short period of time. For sufficiently longer planning times, other settings of the algorithms or gene models may perform better. All combinations of the two gene models GM1 and GM2 and the two repair mechanisms were investigated and the results are based on 100 runs per combination and benchmark in order to obtain meaningful success rates. For every setting, different population sizes in the range of 200 to 600 were used.

4.1 Results of the Heuristics and the Giffler-Thompson Algorithm

The most striking result is that the GTA can solve one benchmark only and this is a simple one with just 50 grid jobs. For an overall comparison, the best result of each benchmark is set to 100%. Table 1 shows the averaged percentages of all benchmarks for all heuristics introduced and the GTA with its priority rules. There are two surprising results: there is only one outstanding procedure and this is not the GTA, but the heuristic *shortest due time*. However, the GTA produces the second best values for the end fitness which gives rise to the hope that its results are good seeds for the initial population of the subsequent EA run. Obviously, our scheduling task differs too much from the pure job shop scheduling problem the GTA is aimed at.

4.2 Best Gene Model and Repair Method

At first, the two gene models are compared in figures 1 and 2 for both repair methods. GM2 performs better in most cases, especially for phenotypic repair, the only exception being the low robustness (cf. Fig. 1) for benchmark kRgA-100. Earlier experiments using a permutation-based coding like the one described in [10] yielded significantly poorer results than GM1. We attribute this to the effect of the segment mutations described in Section 3, as runs without them produced comparably poor results.

Table 1. Results of the heuristics and the Giffler-Thompson algorithm (GTA). The relative raw and end fitness values (abbreviated as *raw f.* and *end f.*) are compared. For the GTA different priority rules were investigated, some of them recommended by [6]. The last two rules were an attempt of tailoring rules to the given problem. They try to keep the grid jobs of one application job together by preferring the scheduling of grid jobs of already begun application jobs.

GTA with seven priority rules and heuristics with different RAS	raw f.	end f.
Giffler-Thompson – longest job first	71 %	3 %
Giffler-Thompson – shortest job first	72 %	2 %
Giffler-Thompson – shortest due time	73 %	10 %
Giffler-Thompson – shortest relative due time	74 %	11 %
Giffler-Thompson – most work remaining	72 %	2 %
Giffler-Thompson – last of considered job set (s-set)	69 %	1 %
Giffler-Thompson – planned application job preferred	68 %	1 %
Shortest due time & RAS-3 (appl. job dependant res. preference)	99 %	86 %
Shortest due time & RAS-2 (cheapest resource always)	98 %	81 %
Shortest due time & RAS-1 (fastest resource always)	96 %	87 %
Shortest working time of grid job & RAS-3	70 %	2 %
Shortest working time of grid job & RAS-2	70 %	2 %
Shortest working time of grid job & RAS-1	70 %	2 %
Shortest working time of appl. job & RAS-3	82 %	6 %
Shortest working time of appl. job & RAS-2	80 %	8 %
Shortest working time of appl. job & RAS-1	79 %	7 %

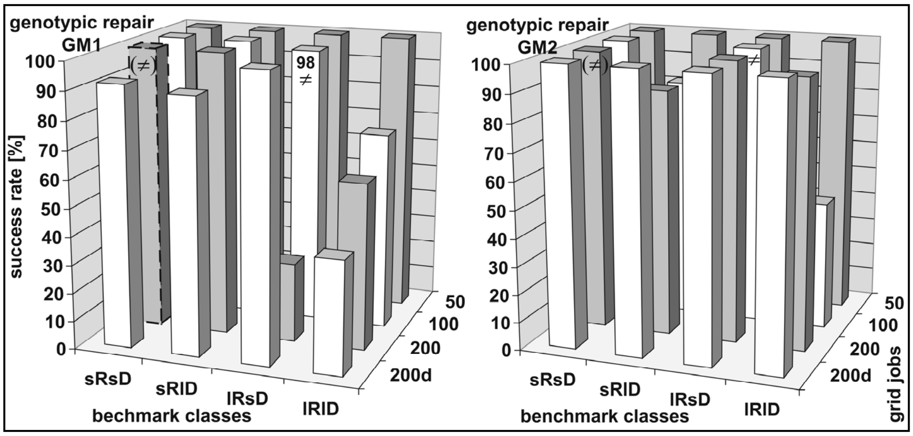


Fig. 1. Comparison of the success rates of the two gene models and genotypic repair. Dotted bars indicate that this result was obtained for one population size only (*low robustness*). If the values are a little below 100%, the exact numbers are given for a better distinction. T-test results of the corresponding fitness values are given for those cases, where significance can not be derived from the success rate directly: ≠: difference significant at 99.9%, (≠): t-test not applicable due to too large variance differences, ≈: differences are not significant.

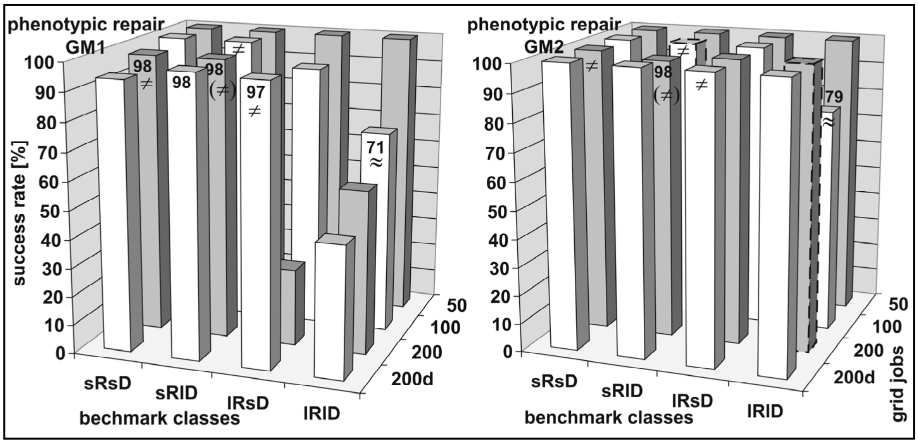


Fig. 2. Comparison of the success rates of the two gene models and phenotypic repair. Explanations see Fig. 1.

An analysis of some runs longer than three minutes reveals the same type of behaviour for more than 50 grid jobs, as shown in Fig. 3. The reduced search space of GM2 allows for a faster progress of the evolution, but cannot produce as good results as GM1 in the long run, because GM1 covers the complete search space. If all other parameters are fixed, the number of grid jobs determines the complexity and, hence the position of the intersection point. For benchmarks of about 50 jobs, this is left of the limit of three minutes and GM1 yields the better results.

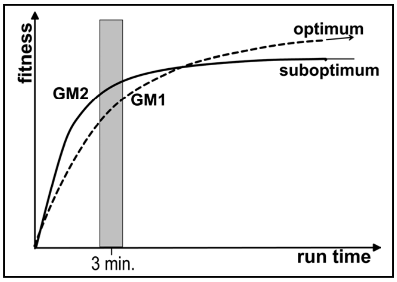
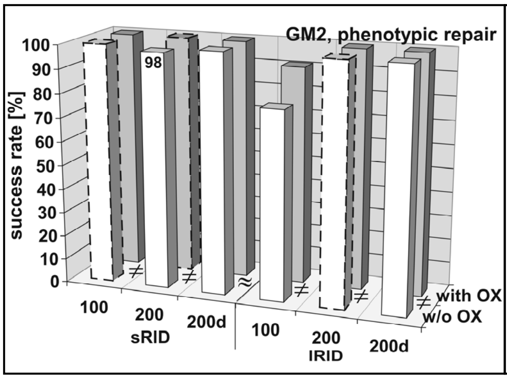


Fig. 3. Basic course of evolution of both gene models for more than 50 grid jobs

A comparison of the two repair methods for GM2 (right parts of Figs. 1 and 2) shows the superiority of phenotypic repair. This can be explained by the greater flexibility in altering the chromosomes. Thus, GM2 and phenotypic repair are used as the basis of the following investigation described in the next two sections.

4.3 Results for the Two Crossover Operators

As the benchmarks of sRlD and lRlD obviously are the most difficult ones for more than 50 grid jobs, further investigations are restricted to them to reduce the effort. The effect of both crossover operators was examined by using them solely or together to either replace or complement the standard operators. The strict PPX operator failed completely, while the OX was beneficial, but only when used alone and in addition to the standard 1- and n-point crossover operators. Fig. 4 shows the results for the added OX operator. They can be explained by the more exploitative character of the OX operator, which completes the standard operators that are more aimed at exploration.



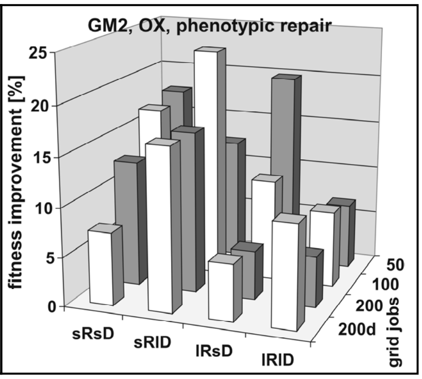


Fig. 4. The effect of the OX operator used in addition to the standard crossover operators. With the exception of sRlD-200d benchmark, all results with and without OX differ significantly. For further explanations, see Fig. 1.

Fig. 5. Improvement of the end fitness compared to the raw fitness of the best heuristic. All differences are significant, as the raw fitness values are far outside of the confidence intervals of the EA results.

The comparison of success rates is completed by the comparison of the averaged fitness differences as shown in Fig. 5 for GM2, phenotypic repair, and the OX crossover. The end fitness values obtained from the best EA runs are better by 5% at the minimum than the raw fitness of the best heuristic. The figure also shows that a small amount of resource alternatives (sR) yields slightly better results than more alternatives. This is due to the fact that more search steps may be required for finding a suitable time slot when more resources are under consideration. This results in fewer evaluations and, hence, a shorter search within the given runtime.

4.4 The Effect of Seeding the Start Population

For the difficult benchmarks *sRlD-200*, *lRlD-200*, and *-200d,* significantly better result can be obtained by seeding the start population with the results of the heuristics, as displayed in Fig. 6. This corresponds to similar improvements of the end fitness not shown here. In the other cases, only minor improvements can be achieved, if any. This means that starting the evolution from randomly generated individuals is sufficient in most cases. Seeding the start population may help in some difficult ones but not in all, as is shown by *lRlD-100.* On the other hand, it makes sense to enter the

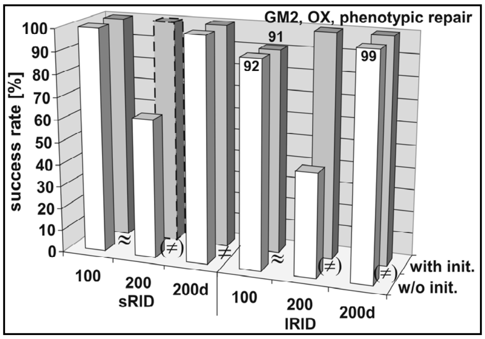


Fig. 6. Seeding the start population with the results of the heuristics yields a significant improvement. For explanations of symbols and dotted bars see Fig. 1.

heuristic results, because only improvements may occur due to the elitist nature of GLEAM. The same experiment was performed using the GTA results, but no significant differences were observed in all cases. This means that the GTA cannot even support the subsequent EA run.

5 Conclusions and Outlook

We have introduced a scheduling problem that is a combination of job scheduling with precedence rules, heterogeneous resources with different performances and costs as well as limited availability, and multi-objective optimisation. To make things worse, a solution is required within a few (here, three) minutes. This combination makes the problem more or less special and unique. In fact, we could not find a similar one in literature. The task is to overcome violations of time and cost limits, which cannot be solved by the heuristics applied in a first step, and to improve the general quality of the schedule. It has been shown that with the limited run time, the restriction of the search space by handing over a part of the search to heuristic resource allocation strategies cannot only eliminate the violations, but also improves the solution quality. Best results have been achieved by a phenotypic repair of precedence violations and the well-known *order-based crossover*, provided that it is applied in addition to the standard 1- and n-point crossover operators. The given problem is so far away from standard job shop scheduling that one of the standard procedures, the Giffler-Thompson algorithm, is not successful.

Our investigation was based on the scenario of planning of new jobs and an empty grid. This is good for testing and tuning algorithms and gene models, but not realistic. The next step will be the application of our system to the situation of replanning due to comparable small alterations like some new resources or a new application job.

References

1. Foster, I., Kesselman, C.: The Anatomy of the Grid: Enabling Scalable Virtual Organisations. Int. J. of Supercomputer Applications 15(3), 200–222 (2001)
2. Brucker, P.: Scheduling Algorithms. Springer, Berlin (2004)
3. Setamaa-Karkkainen, A., Miettinen, K., Vuori, J.: Best Compromise Solution for a New Multiobjective Scheduling Problem. Computers & OR 33(8), 2353–2368 (2006)
4. Brucker, P., Knust, S.: Complex Scheduling. Springer, Berlin (2006)
5. Giffler, B., Thompson, G.L.: Algorithms for Solving Production Scheduling Problems. Operations Research 8, 487–503 (1960)
6. Neumann, K., Morlock, M.: Operations Research. Carl Hanser, München (2002)
7. Süß, W., Quinte, A., Jakob, W., Stucky, K.-U.: Construction of Benchmarks for Comparison of Grid Resource Planning Algorithms. In: Filipe, J., et al. (eds.) Conf. Proc. ICSOFT 2007, vol. PL, Inst.f. Systems and Techn. of Information, Control and Com., pp. 80–87 (2007)
8. Blume, C., Jakob, W.: GLEAM – An Evolutionary Algorithm for Planning and Control Based on Evolution Strategy. In: Conf. Proc. GECCO 2002, Late Breaking Papers (2002)
9. Davis, L. (ed.): Handbook of Genetic Algorithms. V. Nostrand Reinhold, New York (1991)
10. Bierwirth, C., Mattfeld, D.C., Kopfer, H.: On Permutation Representations for Scheduling Problems. In: Voigt, H.-M., et al. (eds.) PPSN IV, vol. 1141, pp. 310–318. Springer, Heidelberg (1996)

Virus Evolution Strategy for Vehicle Routing Problems with Time Windows

Hitoshi Kanoh and Souichi Tsukahara

Department of Computer Science,
Graduate School of Systems and Information Engineering
University of Tsukuba
Tsukuba, Ibaraki, 305-8573, Japan
kanoh@cs.tsukuba.ac.jp, tsukahara@kslab.cs.tsukuba.ac.jp

Abstract. This paper proposes a new solution to the vehicle routing problem with time windows using an evolution strategy adopting viral infection. The problem belongs to the NP-hard class and is very difficult to solve within practical time limits using systematic optimization techniques. In conventional evolution strategies, a schema with a high degree-of-fitness produced in the process of evolution may not be inherited when the fitness of the individual containing the schema is low. The proposed method preserves the schema as a virus and uses it by the infection operation in successive generations. Experimental results using extended Solomon's benchmark problems with 1000 customers proved that the proposed method is superior to conventional methods in both its rates of searches and the probability of obtaining solutions.

Keywords: Vehicle routing problem, NP-hard class, evolution strategy, infection, virus, schema.

1 Introduction

The vehicle routing problem with time windows (VRPTW) is a well-known combinatorial optimization problem that arises in delivery services, logistics, and distribution systems [1, 2]. The problem is how to design a set of minimum-cost vehicle routes originating and terminating at a central depot for a fleet of vehicles that services a set of customers with known demands and time-window constraints.

The VRPTW belongs to the NP-hard class [3] and is very difficult to solve within practical time limits using systematic optimization techniques. Metaheuristics have recently been studied, which includes tabu searches [4, 5], simulated annealing [4, 6, 7], genetic algorithms (GAs) [4, 5, 8, 9], evolution strategies (ESs) [10, 11], and ant-colony optimization [12]. Two-phase hybrid metaheuristics using an evolution strategy has demonstrated especially impressive performance [10]. A (μ, λ)-evolution strategy is used to minimize the number of vehicles in the first search phase. In the second search phase, the total travel distance is minimized with a tabu search algorithm.

In this paper, we propose a method of improving the ES in the two-phase method as the computational cost of the first phase is higher than that of the second. The new method introduces a viral infection into the ES. This is called a virus evolutionary strategy (VES) in this paper. In conventional ESs, a schema with a high degree-of-fitness produced in the process of evolution may not be inherited when the fitness of the individual containing the schema is low. The proposed method preserves the schema as a virus and uses it by the infection operation in successive generations. To use such schema for search is know as a virus GA [13, 14], but no research has yet been published that has reported the introduction of viruses into ESs. While mutations in ESs only carry out local searches, infection greatly changes the chromosomes of the target individual and enables the function of escaping from local optima. The purpose of this study was to improve both the rates of searches and the probability of obtaining solutions in ESs by using infection.

The following sections first describe the objective problem and typical solutions to this. Next, we describe how we generated a virus that was specialized for the VRPTW and the algorithm used in the proposed method in detail. Finally, we present the results of experiments using benchmark problems with 1000 customers [15].

2 Background

2.1 Objective Problem

Figure 1 shows an example of the VRPTW, which consists of a set of identical vehicles (routes), a central depot, and a set of customers. The objective of the VRPTW is to find a set of routes to minimize the number of vehicles needed to supply all customers and the sum of travel time and waiting time for all vehicles under the following constraints. This problem has already been formulated [2, 4].

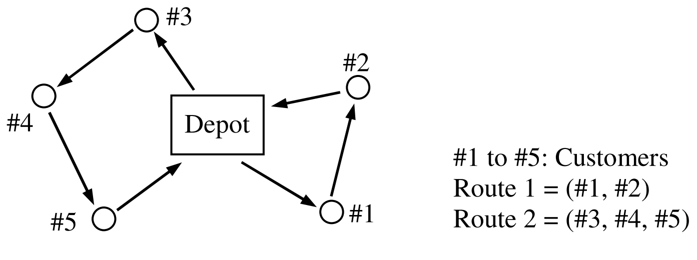


Fig. 1. Example of VRPTW. A separate vehicle is assigned to each of the routes.

- Each customer is only visited once by one of the vehicles.
- Every vehicle has the same capacity, which is greater than or equal to the total demand on the route it travels.
- The number of vehicles is less than or equal to a given number.
- A vehicle arriving at a customer before its time window must wait.
- A solution becomes infeasible if a customer is supplied after its time window has elapsed.
- Each route must start and end within the time window associated with the depot.

2.2 Typical Solutions

Homberger and Gehring proposed the (μ, λ)-ES to minimize the number of vehicles in the first search phase of their two-phase hybrid metaheuristics [10] using mutation and modified Or-opt as follows.

Mutation
The following operations are applied to a randomly selected individual from the population.

- Insertion: Two routes are randomly selected from the individual and a customer selected from one route is inserted along the other route to minimize the travel cost.
- 2-opt: This operation is a kind of crossover in the same individual. First, two routes are randomly selected from the individual. Next, a crossover site is chosen at random and the rear customers are swapped.
- Interchange: This operation interchanges customers once between two randomly selected routes. Each customer is chosen at random and inserted in a place in which it can be located.

Modified Or-opt
This operation is intended to reduce the number of customers along the shortest routes in the individual or even eliminate the shortest routes. All possible attempts are subsequently made to locate all customers on the shortest routes along other routes.

2.3 Virus Theory of Evolution

Nakahara et al. assumed that the cause of evolution is viral infection [16]. Viruses can transfer genes from one organism to target ones, and may change their chromosomes. In neo-Darwinian evolutionary theory, these organisms have been considered to evolve as follows: the organisms' fittest characteristics for survival are produced by mutation, and these characteristics are inherited by their offspring. However, the probability that mutated dominant individuals will appear and increase in nature is very low. Nakahara et al. explained this problem and they insisted that the viral theory of evolution does resolve four points not fully explained by Darwinism: 1) the rapid evolution of the species, 2) the rapid extinction of the species, 3) evolution progressing in a specific direction, and 4) the occurrence of parallel segregation [16].

3 Proposed Method

3.1 Chromosome Representation and Fitness

Each chromosome in the population with the proposed method (VES) represents a candidate solution that consists of a set of routes. Each route can be expressed by a variable-length sequence of customers who should be visited by one vehicle. Let R_j be the j-th route in an individual. Here, $R_j = (C_j(1), \ldots, C_j(n_j))$, $C_j(l)$ is a customer who the vehicle visits to the l-th, and n_j is the number of customers on the route. Figure 2 shows a chromosome representation, where *m* is the number of routes (vehicles) of the individual.

The fitness of an individual can be expressed by the number of vehicles and the total travel time. Individuals in the ordering process of VES are sorted in increasing order of the number of vehicles. If two or more individuals are with the same number of vehicles, they are sorted in increasing order of the total travel time.

R_1 =	$C_1(1)$	…	$C_1(l)$	…	$C_1(n_1)$		
:	:		:				
R_j =	$C_j(1)$	…	$C_j(l)$	…	…	…	$C_j(n_j)$
:	:		:				
R_m =	$C_m(1)$	…	$C_m(l)$	…	…	$C_m(n_m)$	

Fig. 2. Chromosome representation of proposed method. Each individual can be expressed by a two dimensional variable-length array.

3.2 General Procedure

The general procedure for the proposed method is given below, where *P*(*t*) is the population at generation *t* and *Top*(*n*, *P*) is a set of top *n* individuals in population *P*. Mutation and modified Or-opt operation in the procedure have already been reported [10] (see Section 2). The other operations will be described in detail in the following sections.

```
Procedure VES()
  initialize population P(1);
  for(t=1; t<=upper bound of generation; t++){
    empty temporary population P';
    for(i=1; i<=$\lambda$; i++){
      x = select an individual from P(t) at random;
      Mutation(x);
      Modified-Or-opt(x);
      Infection(x) with a probability of $P_{inf}$;
      P' += {x};
    }
    P(t+1) = Top($\mu$-1,P') + Top(1,P(t));
  }
```

3.3 Initial Population

The initial population in this paper is generated using Homberger and Gehring's method of insertion [10]. The heuristics for inserting customers into a route for the VRPTW was introduced by Solomon [1].

Heuristics. The feasibility of inserting a customer into a route is checked by inserting the customer between all the links on the current route and then selecting the link that has the lowest additional cost of insertion. A feasibility check is done to examine all the constraints including time windows and load capacity. Only feasible insertions will be accepted. When the current route is full without any new customers being accepted, the heuristics will start a new route.

The procedure for generating the initial population with the proposed method involves seven steps.

[Step1] Select a customer at random and generate a route which consists only of that customer.
[Step2] Choose either clockwise or counterclockwise rotation at random, and select the customer who is in that direction.
[Step3] Insert the customer into the route according to the heuristics.
[Step4] Repeat Steps 2 and 3 until all the customers have been routed.
[Step5] Let this set of routes be an individual.
[Step6] Repeat Steps 1 to 5 μ times.
[Step7] Let this set of individuals be an initial population.

3.4 Virus Population

There may be two or more routes where a customer visited first are the same and a customer visited last are the same on all the routes in the population. In the case in Fig. 3, there are relations $C_1(1) = C_2(1)$ and $C_1(4) = C_2(3)$ between R_1 and R_2. Let a route where the number of customers is maximum among them be a virus (e.g., R_1). When the number of customers is the same, let a route with the shortest travel time be a virus. We regard all the viruses found until the current generation as a population of viruses. The viruses are updated whenever a new route is produced by mutation or modified Or-opt. Each virus consists of one route, while an individual consists of a set of routes that includes all customers.

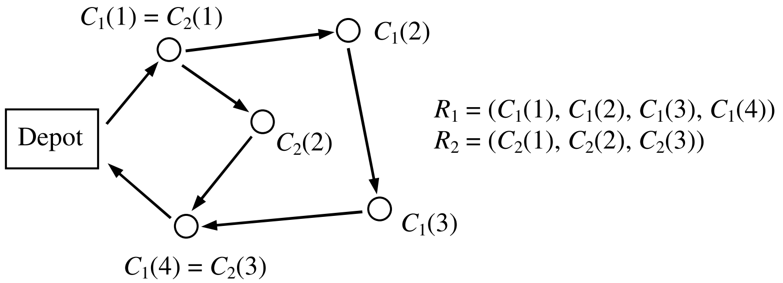

$C_1(1) = C_2(1)$
$C_1(2)$
Depot
$C_2(2)$
$R_1 = (C_1(1), C_1(2), C_1(3), C_1(4))$
$R_2 = (C_2(1), C_2(2), C_2(3))$
$C_1(3)$
$C_1(4) = C_2(3)$

Fig. 3. Example of process of generating virus population

The procedure for updating the virus is given below, where $V_k = (C_k(1), \dots, C_k(n_k))$ is the k-th virus in the population of viruses, $V = \{V_k \mid k=1, \dots, N_{virus}\}$ is the population of viruses, and N_{virus} is the number of viruses (this is a variable). Here, $T(V_k)$ is the travel time on V_k and $R_j = (C_j(1), \dots, C_j(n_j))$ is a newly created route.

```
Procedure update-of-virus()
  for($k$=1; $k$<=$N_{virus}$; $k$++){
    $N'$ = $N_{virus}$;
    if($C_j$(1)=$C_k$(1) and $C_j$($n_j$)=$C_k$($n_k$)){
      if($n_j$>$n_k$ or ($n_j$=$n_k$ and $T$($R_j$)<$T$($V_k$))) {$V$=$V$-{$V_k$}+{$R_j$};}
    }
    else {$V$=$V$+{$R_j$}; $N'$++; }
  }
  $N_{virus}$ = $N'$;
```

Table 1 lists an example of the population of viruses. A blank table for all combinations of the first and last visited customers (for 10 customers, the combinations are (#1, #2), (#1,#3), (#1, #4), …, (#9, #10)) is prepared at the beginning of the VES procedure, and then a new virus is written in the blank or over the old virus by using the update-of-virus procedure. When the number of customers N_{cus} is 1000, the linage in this table is $N_{cus}(N_{cus}-1)/2 < 5\times10^5$ (this is not so large). No additional cost is incurred in maintaining this table, because the computational cost of the update-of-virus procedure can be negligibly small as compared with that for creating a new route.

Table 1. Example of virus population in 10 customers. Each line corresponds to a virus. The hash mark "#" means the serial number of customers. $C(1)$ and $C(n_k)$ are respectively the first and the last visited customers, and n_k is the number of customers with the virus.

$C(1)$	$C(n_k)$	n_k	Travel time	Route
#1	#2			
#1	#3	6	162	(#1, #4, #6, #2, #10, #3)
#1	#4			
#1	#5	5	93	(#1, #8, #7, #2, #5)
…	…	…	…	…
…	…	…	…	…
#9	#10	5	125	(#9, #3, #7, #8, #10)

3.5 Infection

Individuals are infected after the modified Or-opt operation with an infection probability of P_{inf}. The procedure is as follows.

```
Procedure Infection()
  Select a route $R_j$ from the individual randomly;
  for($k$=1; $k$<=$N_{virus}$; $k$++)
    if($C_j$(1)=$C_k$(1) and $C_j$($n_j$)=$C_k$($n_k$))
      if($n_j$<$n_k$ or ($n_j$=$n_k$ and $T$($R_j$)>$T$($V_k$))){
        $R_j$ is replaced by $V_k$;
        Move or delete customers; /* see follows */
      }
```

The operation “move or delete customers” in the above procedure is done as follows: if there are customers that are included in V_k and not included in R_j, then delete those customers from the other routes, and if there are customers that are not included in V_k and included in R_j, then insert those customers into the other routes so that the total length of these routes can be minimized.

4 Experiments

4.1 Experimental Method

To evaluate how well the proposed method performed, we conducted two experiments using benchmark problems with 1000 customers (extended Solomon's VRPTW instances [15]). The problems were classified into six types according to the distribution of customers and time-window constraints as listed in Table 2. All six problems had three parts: the type of problem, the number of customers divided by 100, and the sequential number (e.g., c1_10_1). Homberger and Gehring minimized the number of vehicles (routes) by obtaining the ES and the total travel cost using a tabu search [10]. Here, we only tried to minimize the number of vehicles by using the VES, as the purpose of this study was to improve the ES. In addition, the experiments were done under conditions of μ=10 and λ=20, using a 3.2-GHz Pentium IV PC.

Table 2. Types of benchmark problems. The problems are classified into six types according to the distribution of customers and time-window constraints.

Distribution of customers	Time-window constraints	
	Tight	Loose
Cluster	C1	C2
Uniform	R1	R2
Mix	RC1	RC2

4.2 Experimental Results

We first compared the computational time for the proposed method with infection (VES) and without infection (ES). The time required for the number of vehicles to decrease to N_{10} was measured for 12 benchmark problems. Table 3 lists the average of 10 trials for these indicated by *Time*. In this table, the ratio means (*Time*(ES)-*Time*(VES))/*Time*(ES), N_{10} is the number of vehicles definitely obtained through the 10 trials (the worst value in the best value for each trial), and N_{init} is the best value in the initial population of VES. Here, N_{ref} is the best value found by 2007 [15]. The infection probability, P_{inf} = 25%, was determined by conducting preliminary experiments. We can see from the table that the computational times were improved in 10 of the 12 problems by using viral infection. Problem r2_10_2 is a singularly bad case, since as a nearly best value had already been found in the initial population (i.e., N_{init}=20), no infection was necessary.

Table 3. Computational time required for number of vehicles to be decreased to N_{10}. The ratio means (*Time*(ES)-*Time*(VES))/*Time*(ES), N_{init} is the best value in the initial population of VES, and N_{ref} is the best value found by 2007.

Problem	*Time* (sec)		Ratio (%)	Number of vehicles		
	ES	VES		N_{10}	N_{init}	N_{ref}
c1_10_1	164	143	12	103	129	100
c1_10_2	110	52	53	109	120	91
c2_10_1	483	322	33	33	41	30
c2_10_2	325	291	10	34	40	29
r1_10_1	17	15	10	101	105	100
r1_10_2	22	13	43	94	96	91
r2_10_1	175	112	36	21	23	19
r2_10_2	87	209	-140	19	20	19
rc1_10_1	98	99	-2	96	103	90
rc1_10_2	36	22	37	93	97	90
rc2_10_1	398	188	53	26	28	21
rc2_10_2	227	160	29	25	26	18

Table 4. Frequency distribution of number of vehicles in 100 trials for problem of C1_10_1 (N_{ref} = 100)

Number of vehicles	Infection probability				
	0%	25%	50%	75%	100%
100	26	57	71	85	83
101	40	36	27	12	16
102	20	6	2	2	1
103	11	1	0	1	0
104	3	0	0	0	0

Table 5. Frequency distribution of number of vehicles in 100 trials for problem of R1_10_1 (N_{ref} = 100)

Number of vehicles	Infection probability				
	0%	25%	50%	75%	100%
100	92	97	96	96	98
101	8	3	4	4	2

Table 6. Frequency distribution of number of vehicles in 100 trials for problem of RC1_10_1 ($N_{ref} = 90$)

Number of vehicles	Infection probability				
	0%	25%	50%	75%	100%
92	0	2	0	3	5
93	12	23	22	30	32
94	47	40	52	43	37
95	29	25	19	19	22
96	12	10	6	5	2
97	0	0	1	0	2

Next, the number of vehicles in the best individual in the population at the 10000-th generation was evaluated using the problems for types C1, R1, and RC1 to investigate the probability of obtaining a good solution. Tables 4 to 6 list their frequency distributions in 100 trials when $P_{inf} = 0$, 25, 50, 70, and 100%. The best value for C1 and R1 (i.e., $N_{ref} = 100$) was obtained and the probability of obtaining the best value was highest when $P_{inf} = 75$ or 100% for C1 and 100% for R1. The best value for RC1 (i.e., $N_{ref} = 90$), on the other hand, was not obtained. However, comparing the frequency where good solutions were obtained (e.g., the number of vehicles is less than 94), the frequencies were larger when $P_{inf} > 0$ than when $P_{inf} = 0$. These results prove the effectiveness of viral infection in VRPTWs.

5 Conclusions

We proposed a new solution to VRPTWs with an ES adopting viral infection. Experimental results using extended Solomon's benchmark problems with 1000 customers proved that the proposed ES is superior to conventional ESs in both its rates of searches and the probability of obtaining good solutions. The present method promotes evolution by using partial solutions obtained in the process of evolution. Infection plays a role in building schemata that mutation does not, and infection does not need any complicated chromosome representations to avoid lethal genes that crossover needs. Consequently, the combination of infection and mutation can be useful for a wide variety of NP-hard problems.

In the benchmark problems used, the travel distance between two arbitrary customers was fixed, but in practical problems, this will change with time because of congestion due to traffic. We next intend to apply the new method to practical problems with a road map database and actual traffic data.

References

1. Solomon, M.M.: Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research 35(2), 254–265 (1987)
2. The VRP Web, `http://neo.lcc.uma.es/radi-aeb/WebVRP/`

3. Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity of Vehicle Routing and Scheduling Problems. Networks 11, 221–227 (1981)
4. Tan, K.C., Lee, L.H., Ou, K.: Artificial Intelligence Heuristics in Solving Vehicle Routing Problems with Time Window Constraints. Engineering Applications of Artificial Intelligence 14, 825–837 (2001)
5. Ropke, S., Pisinger, D.: A General Heuristic for Vehicle Routing Problems, Technical Report, Department of Computer Science, University of Copenhagen (2005)
6. Thangiah, S.R., Osman, I.H., Sun, T.: Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows, Technical Report 27, Computer Science Department, Slippery Rock University PA USA (1994)
7. Bent, R., Hentenryck, P.V.: A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. Transportation Science 38(4), 515–530 (2004)
8. Bräysy, O.: Genetic Algorithms for the Vehicle Routing Problem with Time Windows. Special Issue on Bioinformatics and Genetic Algorithms, 33–38 (2001)
9. Berger, J., Barkaoui, M.: A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. Computers & Operations Research 31, 2037–2053 (2004)
10. Homberger, J., Gehring, H.: A Two-Phase Hybrid Metaheuristic for the Vehicle Routing Problem with Time Windows. European Journal of Operational Research 162, 220–238 (2005)
11. Mester, D., Bräysy, O.: Active Guided Evolution Strategies for Large-Scale Vehicle Routing Problems with Time Windows. Computers & Operations Research (2005)
12. Gambardella, L.M., Taillard, E., Agazzi, G.: MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, New Ideas in Optimization, pp. 63–76. McGraw-Hill, London (1999)
13. Kanoh, H., Matsumoto, M., Hasegawa, K., Kato, N., Nishihara, S.: Solving Constraint Satisfaction Problems by a Genetic Algorithm adopting Viral Infection. International Journal on Engineering Applications of Artificial Intelligence 10(6), 531–537 (1997)
14. Kanoh, H.: Dynamic Route Planning for Car Navigation Systems using Virus Genetic Algorithms. International Journal of Knowledge-Based and Intelligent Engineering Systems 11(1), 65–78 (2007)
15. Homberger, J.: Benchmarks - Vehicle Routing and Traveling Salesperson Problems, `http://www.sintef.no/static/am/opti/projects/top/vrp/`
16. Nakahara, H., Sagawa, T., Fuke, T.: Virus Theory of Evolution. Bulletin of Yamanashi Medical College 3, 14–18 (1986)

Learning Fuzzy Rules with Evolutionary Algorithms — An Analytic Approach

Jens Kroeske[1], Adam Ghandar[2], Zbigniew Michalewicz[3], and Frank Neumann[4]

[1] School of Mathematics, University of Adelaide, Adelaide, SA 5005, Australia
[2] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia
[3] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia; also at Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland, and Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland
[4] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

Abstract. This paper provides an analytical approach to fuzzy rule base optimization. While most research in the area has been done experimentally, our theoretical considerations give new insights to the task. Using the symmetry that is inherent in our formulation, we show that the problem of finding an optimal rule base can be reduced to solving a set of quadratic equations that generically have a one dimensional solution space. This alternate problem specification can enable new approaches for rule base optimization.

1 Introduction

Fuzzy rule based solution representations combined with evolutionary algorithms are a powerful real world problem solving technique, for example see [2,5,9,15,18,19]. Fuzzy logic provides benefits in naturally representing real world quantities and relationships, fast controller adaptation, and a high capacity for solutions to be interpreted. The typical scenario involves using an evolutionary algorithm to find optimum rule bases with respect to some application specific evaluation function, see [7,11,13].

A fuzzy rule is a causal statement that has an *if-then* format. The *if* part is a series of conjunctions describing properties of some linguistic variables using fuzzy sets that, if observed, give rise to the *then* part. The *then* part is a value that reflects the consequence given the case that the *if* part occurs in full. A rule base consists of several such rules and is able to be evaluated using fuzzy operators to obtain a value given the (possibly partial) fulfilment of each rule.

Membership functions are a crucial part of the definition as they define the mappings to assign meaning to input data. They map crisp input observations of linguistic variables to degrees of membership in some fuzzy sets to describe properties of the linguistic variables. Suitable membership functions are designed depending on the specific characteristics of the linguistic variables as well as peculiar properties related to their use in optimization systems. Triangular membership functions are widely used primarily for the reasons described in [16].

Other common mappings include ‘gaussian’ [11] and ‘trapezoidal’ [8] membership functions. The functions are either predefined or determined in part or completely during an optimization process. A number of different techniques have been used for this task including statistical methods [7], heuristic approaches [2], and genetic and evolutionary algorithms [5,9,14,18]. Adjusting membership functions during optimization is discussed in [9,20].

A financial computational intelligence system for portfolio management is described in [7]. Fuzzy rule bases are optimized in an evolutionary process to find rules for selecting stocks to trade. A rule base that could be produced using this system could look as follows:

– If *Price to Earnings Ratio* is *Extremely Low* then *rating* = 0.9
– If *Price Change* is *High* and *Double Moving Average Sell* is *Very High* then *rating* = 0.4

The *if* part in this case specifies some financial accounting measures (Price to Earnings ratio) and technical indicators [1] used by financial analysts; the output of the rule base is a combined rating that allows stocks to be compared relative to each other. In that system rule bases were evaluated in the evolutionary process using a function based on a trading simulation.

The task of constructing rule base solutions includes determining rule statements, membership functions (including the number of distinct membership sets and their specific forms) and possible outputs. These parameters and the specification of data structures for computational representation have a significant impact on the characteristics and performance of the optimization process. Previous research in applications [8,17,1] has largely consisted and relied on intuition and experimental analysis for designs and parameter settings. This paper takes a theoretical approach to the analysis of a specific design of a fuzzy rule base optimization system that has been used in a range of successful applications [6,7,11,13]; we utilize the symmetry that is inherent in the formulation to gain insight into the optimization. This leads to an interesting alternate viewpoint of the problem that may in turn lead to new approaches.

In particular, our formal definition and framework for the fuzzy rule base turns the optimization problem into a smooth problem that can be analyzed analytically. This analysis reduces the problem to a system of quadratic equations whose solution space has the surprising property that it generically contains a whole line. It should be possible to utilize this fact in the construction of fast and efficient solvers, which will be an important application of this research. The approach in this paper builds on experimental research presented in [7,6], but it should be noted that a number of other mechanisms have been proposed for encoding fuzzy rules [8].

The methods we consider could be used in an evaluation process where the error is minimized with respect to fitting rule bases to some training data — in the context of the above example this would allow a system to learn rules with an output that is directly calculated from the data. For example a rule base evaluated in this way could be used to forecast the probability that a stock has positive price movement [10,12] in some future time period. A rule in such a rule base could look like: If *Price to Earnings Ratio* is *Extremely Low* and *Double Moving Average Buy* is *Very High* then *probability of positive price movement* is 0.75. In this case the training data set would be some historical stock market data similar to that used in [6,7].

The structure of this paper is as follows: Section 2 contains the formal definitions for the analysis presented in Section 3. Section 4 concludes the paper.

2 Approach

In this section we introduce the formulation of the models used in the analysis, including the rule base solution representation, the rule base interpretation method and the evaluation function.

2.1 Rule Base Solution Representation and Interpretation

Let us introduce some precise definitions of what is meant by the rule base solution representation. First of all, we are given L linguistic variables $\{A^1, ..., A^L\}$. Each linguistic variable A^i has M_i linguistic descriptions $\{A^i_1, ..., A^i_{M_i}\}$ that are represented by triangular membership functions μ^i_j, $j = 1, ..., M_i$. A fuzzy rule has the form

$$\text{If } A^{i_1} \text{ is } A^{i_1}_{j_1} \text{ and } A^{i_2} \text{ is } A^{i_2}_{j_2} \text{ and } \cdots \text{ and } A^{i_k} \text{ is } A^{i_k}_{j_k} \text{ then } o, \tag{1}$$

where $i_1, ...i_k \in \{1, ..., L\}$, $j_k \in \{1, ..., M_{i_k}\}$ and $o \in [0, 1]$.

A rule base is a set of several rules. Let us assume that we are given a rule base consisting of n rules:

$$\begin{array}{lll}
\text{If } A^{i^1_1} \text{ is } A^{i^1_1}_{j^1_1} \text{ and } A^{i^1_2} \text{ is } A^{i^1_2}_{j^1_2} \text{ and } \cdots \text{ and } A^{i^1_{k_1}} \text{ is } A^{i^1_{k_1}}_{j^1_{k_1}} & \text{then} & o^1 \\
\text{If } A^{i^2_1} \text{ is } A^{i^2_1}_{j^2_1} \text{ and } A^{i^2_2} \text{ is } A^{i^2_2}_{j^2_2} \text{ and } \cdots \text{ and } A^{i^2_{k_2}} \text{ is } A^{i^2_{k_2}}_{j^2_{k_2}} & \text{then} & o^2 \\
\qquad\qquad\qquad\qquad\vdots & & \vdots \\
\text{If } A^{i^n_1} \text{ is } A^{i^n_1}_{j^n_1} \text{ and } A^{i^n_2} \text{ is } A^{i^n_2}_{j^n_2} \text{ and } \cdots \text{ and } A^{i^n_{k_n}} \text{ is } A^{i^n_{k_n}}_{j^n_{k_n}} & \text{then} & o^n,
\end{array}$$

where $i^m_l \in \{1, ..., L\}$ and $j^m_l \in \{1, ..., M_{i^m_l}\}$. Given a vector $x \in \mathbb{R}^L$ of observed values, whose components are values for the linguistic variables $A^1, ..., A^L$, we can evaluate the rule base as follows: the function ρ describes the way the rule base interprets data observations x to produce a single output value. This value has an application specific meaning and can be taken to be a real number (usually normalized to lie between zero and one). More precisely, ρ is defined as follows:

$$\rho : \mathbb{R}^L \to \mathbb{R}$$

$$x = \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^L \end{pmatrix} \mapsto \frac{\sum_{m=1}^n o^m \prod_{l=1}^{k_m} \mu^{i^m_l}_{j^m_l}(x^{i^m_l})}{\sum_{m=1}^n o^m}.$$

2.2 Evaluation Function

We consider an evaluation function (to minimize) that measures the error when training a rule base to fit a given data set. This *training data* consists of a set $\{x_i, y_i\}_{i=1...N}$, where each

$$x_i = \begin{pmatrix} x_i^1 \\ x_i^2 \\ \vdots \\ x_i^L \end{pmatrix}$$

is a vector that has as many components as there are linguistic variables, i.e. $x_i \in \mathbb{R}^L \; \forall \; i = 1, ..., N$, and each y_i is a real number, i.e. $y_i \in \mathbb{R} \; \forall \; i = 1, ..., N$. Then the evaluation function has the form

$$\epsilon = \sum_{i=1}^{N} (\rho(x_i) - y_i)^2 \tag{2}$$

$$= \sum_{i=1}^{N} \left(\frac{\sum_{j=1}^{n} a_{ij} o^j}{\sum_{j=1}^{n} o^j} - y_i \right)^2 , \tag{3}$$

where

$$a_{sm} = \prod_{l=1}^{k_m} \mu_{j_l^m}^{i_l^m}(x_s^{i_l^m}).$$

Our aim is to optimize the rules base in such a way that the evaluation function ϵ becomes minimal. This involves two separate problems. Firstly, the form of the membership functions μ_j^i may be varied to obtain a better result. Secondly, the rule base may be varied by choosing different rules or by varying the weights o^i. In this paper we will concentrate on the second problem, taking the form of the membership functions to be fixed. For example, we can standardize the number of membership functions for each linguistic variable A^i to be $M_i = 2n_i - 1$ and define

$$\mu_j^i = \begin{cases} 0 & : \; x \leq \frac{j-1}{2n_i} \\ 2n_i x + 1 - j & : \; x \in \left[\frac{j-1}{2n_i}, \frac{j}{2n_i}\right] \\ -2n_i x + 1 + j & : \; x \in \left[\frac{j}{2n_i}, \frac{j+1}{2n_i}\right] \\ 0 & : \; x \geq \frac{j+1}{2n_i} \end{cases}$$

for $j = 1, ..., 2n_i - 1 = M_i$. These functions are shown in Figure 1.

Moreover, we can consider the number n of rules to be fixed by either working with a specific number of rules that we want to consider, or by taking n to be the number of all possible rules (this number will be enormous, but each rule whose optimal weight is zero, or sufficiently close to zero can just be ignored and most weights will be of that form), depending on the application. The resulting optimization problem will be considered in 3.2.

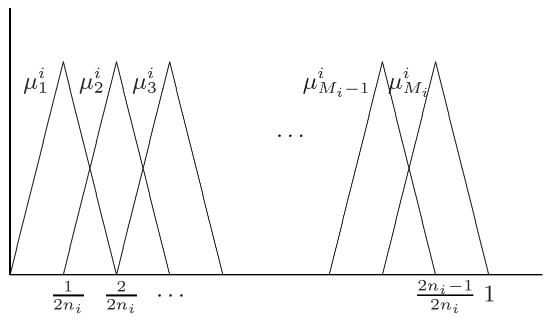


Fig. 1. Membership Functions

3 Analysis

This section contains the detailed analysis of the problem described in Section 2. We firstly determine the maximum possible number of rules and then consider the optimization problem for the evaluation function. As a result, we are able to reduce the optimization problem to a system of equations (6), that has the remarkable property that it allows (generically) a one-dimensional solution space. This is the content of Theorem 1.

3.1 Search Space

The search space is the set of all potential rule base solutions. Let us first of all compute the maximum number of rules $n_{\max}$ that we can have. Each rule can be written in the form

$$\text{If } A^1 \text{ is } A^1_{j_1} \text{ and } A^2 \text{ is } A^2_{j_2} \text{ and } \cdots \text{ and } A^L \text{ is } A^L_{j_L} \text{ then } o,$$

where in this case $j_i \in \{0, 1, ..., M_i\}$ and $j_i = 0$ implies that the linguistic variable A^i does not appear in the rule. Then we have

$$n_{\max} = (M_1 + 1) \times (M_2 + 1) \times \cdots \times (M_L + 1) - 1.$$

Note that we have subtracted 1 to exclude the empty rule. If we include the possible choices of weights o^i with discretization $o^i \in \{0, \frac{1}{d}, ..., 1\}$, then we have a system of

$$(d+1)^{n_{\max}}$$

possible rule bases.

3.2 Optimization Problem

In this subsection we will treat the optimization problem described in 2.2. We have to take the training data $\{x_i, y_i\}_{i=1\dots N}$ and the various membership functions μ^i_j as given, so we can treat the various a_{ij} as constants and simplify

$$
\begin{aligned}
\epsilon(o) &= \sum_{i=1}^{N} \left(\frac{\sum_{j=1}^{n} a_{ij} o^j}{\sum_{j=1}^{n} o^j} - y_i \right)^2 \\
&= \sum_{i=1}^{N} \left(\frac{\sum_{j=1}^{n} (a_{ij} - y_i)^2 o^j o^j + 2 \sum_{j<k} (a_{ij} - y_i)(a_{ik} - y_i) o^j o^k}{\left(\sum_{j=1}^{n} o^j\right)^2} \right) \\
&= \frac{\sum_{j=1}^{n} A_{jj} o^j o^j + 2 \sum_{j<k} A_{jk} o^j o^k}{\left(\sum_{j=1}^{n} o^j\right)^2} \\
&\quad \text{with } \; A_{jk} = \sum_{i=1}^{N} (a_{ij} - y_i)(a_{ik} - y_i) \\
&= \frac{\sum_{j=1}^{n} \sum_{k=1}^{n} A_{jk} o^j o^k}{\left(\sum_{j=1}^{n} o^j\right)^2}.
\end{aligned}
$$

We want to find weights o^i such that this expression becomes minimal. In our formulation this requirement is smooth in the o^i, so we can compute the partial derivatives of the evaluation function with respect to the weights. At a minimal point $o_{\min} \in \mathbb{R}^n$, we must have

$$\frac{\partial \epsilon}{\partial o^1}(o_{\min}) = 0, \frac{\partial \epsilon}{\partial o^2}(o_{\min}) = 0, ..., \frac{\partial \epsilon}{\partial o^n}(o_{\min}) = 0.$$

It will turn out that this requirement is equivalent to a system of quadratic equations. So let us compute

$$
\begin{aligned}
\frac{\partial \epsilon}{\partial o^q}(o) &= 2 \frac{\left(\sum_{i=1}^{n} A_{iq} o^i\right) \left(\sum_{k=1}^{n} o^k\right) - \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} o^i o^j}{\left(\sum_{i=1}^{n} o^i\right)^3} \qquad (4) \\
&= \frac{2}{\left(\sum_{i=1}^{n} o^i\right)^3} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} (A_{iq} - A_{ij}) o^i o^j \right). \qquad (5)
\end{aligned}
$$

If we can simultaneously solve these n equations

$$\frac{\partial \epsilon}{\partial o^1}(o) = 0, \frac{\partial \epsilon}{\partial o^2}(o) = 0, ..., \frac{\partial \epsilon}{\partial o^n}(o) = 0,$$

then we have found a local extrema. For only two rules, for example, we obtain

$$
\begin{aligned}
\frac{\partial \epsilon}{\partial o^1}(o) &= \frac{2 o^2}{\left(o^1 + o^2\right)^3} \left((A_{11} - A_{12}) o^1 + (A_{21} - A_{22}) o^2 \right) \\
\frac{\partial \epsilon}{\partial o^2}(o) &= \frac{2 o^1}{\left(o^1 + o^2\right)^3} \left((A_{12} - A_{11}) o^1 + (A_{22} - A_{21}) o^2 \right)
\end{aligned}
$$

Therefore, if we assume that $o^1 \neq 0$ or $o^2 \neq 0$, then the optimal solution is

$$o^1 = \frac{A_{22} - A_{21}}{A_{11} - A_{12}} o^2.$$

This is a whole line that intersects zero in $\mathbb{R}^2$. This phenomena can be seen clearly in the following picture:

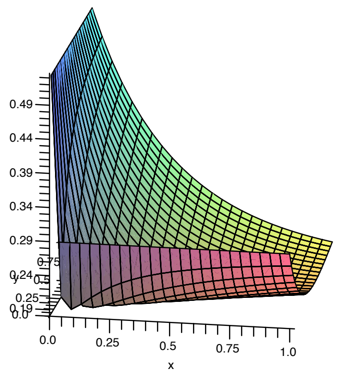


Fig. 2. Evaluation function for two rules

More than Two Rules. If we have more than two rules, then the conditions become

$$\frac{\partial \epsilon}{\partial o^q} = 0 \Leftrightarrow \left(\sum_{i=1}^{n} \sum_{j=1}^{n} (A_{iq} - A_{ij}) o^i o^j \right) = 0, \; q = 1, ..., n. \tag{6}$$

Theorem 1. *Generically, there exists a one-parameter family of solutions to the system (6). Hence the space of extremal points for ϵ is a line in $\mathbb{R}^n$ that passes through zero.*

Proof. We will show that the n equations (6) are dependent, i.e. that we only need to solve $n-1$ of these equations and the n-th equation then follows automatically. For this purpose, we rewrite the system

$$\begin{aligned} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} (A_{iq} - A_{ij}) o^i o^j \right) &= \sum_{\substack{j=1 \\ j \neq q}}^{n} o^j \underbrace{\left((A_{qq} - A_{qj}) o^q + (A_{jq} - A_{jj}) o^j \right)}_{B_{qj}} \\ &\quad + \sum_{\substack{j=1 \\ j \neq q}}^{n} \sum_{\substack{i=1 \\ i \notin \{q,j\}}}^{n} \left((A_{iq} - A_{ij}) o^i o^j \right). \end{aligned}$$

Note that

$$B_{qj} = -B_{jq}.$$

Denote the q-th equation by E_q. Using the equality above, we compute

$$\begin{aligned}\sum_{k=1}^{n} o^k E_k &= \sum_{k=1}^{n} \sum_{\substack{j=1 \\ j \neq q}}^{n} \underbrace{B_{kj} o^k o^j}_{=0} \\ &\quad + \sum_{k=1}^{n} \sum_{\substack{j=1 \\ j \neq q}}^{n} \sum_{\substack{i=1 \\ i \notin \{q,j\}}}^{n} \left(\underbrace{(A_{ik} - A_{ij})}_{C_{ijk}} o^i o^j o^k \right) \\ &= 0.\end{aligned}$$

The last term vanishes due to the fact that the tensor C_{ijk} is symmetric in the index pair (i, j), symmetric in the index pair (i, k) and skew (i.e. anti-symmetric) in the index pair (j, k). Such a tensor has to vanish identically. It is hence sufficient to solve (6) just for $(n-1)$ equations, the last equation is automatically satisfied. $\square$

Remark. Every local extrema of the function ϵ is in fact a local minima.

Proof. Assume $o_{\max}$ is a local maxima. There exists a hyperplane $E \subset \mathbb{R}^n$ through $o_{\max}$ such that $\left(\sum_i o^i\right)^2$ is constant on E. Now $A = (A_{jk})$ can be written as

$$A = K^T K \text{ with } K_{ij} = a_{ij} - y_i$$

and is hence positive definite. Therefore $A_{ij} o^i o^j$ has a unique local extrema which is a minima. This shows that there is a direction $l \subset E \subset \mathbb{R}^n$ such that $A_{ij} o^i o^j$ increases on l. But then ϵ increases on l as well. This is a contradiction. $\square$

Remark. The analysis in this subsection is independent of the form of the membership functions, in particular they need not be triangular as in 2.2.

4 Conclusions and Future Work

We have successfully reduced the problem of finding optimal weights o^i for a rule base (given an arbitrary set of training data points) to a system of n equations for n unknowns, where n is the number of rules. Moreover, we have shown that the space of extremal points for the evaluation function is a line through the origin in $\mathbb{R}^n$. Hence a genetic algorithms will be able to find an optimal solution in $[0,1]^n$ using well-established and fast methods [3,4]. The reason for this, somewhat surprising, result lies in the specific form of our rule base formulation: not the values of the weights themselves are important, but the relationship that they have with respect to each other. Mathematically speaking, the optimal solution o is really an element of $(n-1)$-dimensional projective space $\mathbb{RP}^{n-1}$, rather that an element of $\mathbb{R}^n$. It should be noted that the t-norm and defuzzification method used in this analysis produces smooth functions, using other operators such as the min t-norm would make the analytic treatment more problematic.

It is possible to use the analysis in this paper to design an optimization process in which combinations of rules consisting of the *if* parts are selected and then evaluated using a fast algorithm to find the optimal *then* parts (output weights) to produce a rule base. This would be beneficial in a range of applications as mentioned in the introduction. For example, reducing the size of the search space by removing the assignment of output weights from the general rule base search problem; or by fixing an initial structure for the rules (using knowledge from the application domain or because of specific application requirements) that feasible solutions should contain and then redefining the search objective to extend this set rather than using a free search — this is for instance a very useful feature in financial applications [7]. As a part of this further research, we will also examine, combinatorially and empirically, algorithms and genotype representations that utilize the reduction in complexity that arises from the analysis in this paper.

References

1. Allen, F., Karjalainen, R.: Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51, 245–271 (1999)
2. Bai, A.: Determining fuzzy membership functions with tabu search an application to control. Fuzzy Sets and Systems 139(1), 209–225 (2003)
3. Courtois, N., Goubin, L., Meier, W., Tacier, J.-D.: Solving underdefined systems of multivariate quadratic equations. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 211–227. Springer, Heidelberg (2002)
4. Davis, L. (ed.): Handbook of genetic algorithms (1991)
5. Esmin, A., Lambert-Torres, G.: Evolutionary computation based fuzzy membership functions optimization. In: IEEE International Conference on Systems, Man and Cybernetics ISIC, pp. 823–828. IEEE, Los Alamitos (2007)
6. Ghandar, A., Michalewicz, Z., Schmidt, M., To, T.-D., Zurbruegg, R.: A computational intelligence portfolio construction system for equity market trading. In: CEC 2007: Proceedings of the 2007 IEEE Congress on Evolutionary Computation. IEEE Press, Los Alamitos (2007)
7. Ghandar, A., Michalewicz, Z., Schmidt, M., To, T.-D., Zurbruegg, R.: Computational intelligence for evolving trading rules. IEEE Transactions On Evolutionary Computation, Special Issue on Evolutionary Computation for Finance and Economics (to appear, 2008)
8. Gomez, J.: Evolution of fuzzy rule based classifiers. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1150–1161. Springer, Heidelberg (2004)
9. Guo, N.R., Li, T.-H.S., Kuo, C.-L.: Design of hierarchical fuzzy model for classification problem using gas. Comput. Ind. Eng 50(1), 90–104 (2006)
10. Jegadeesh, N., Titman, S.: Profitability of momentum strategies: An evaluation of alternative explanations 56(2), 699–720 (2001)
11. Johnson, R., Melich, M., Michalewicz, Z., Schmidt, M.: Coevolutionary optimization of fuzzy logic intelligence for strategic decision support. IEEE Transactions on Evolutionary Computation 9(6), 682–694 (2005)
12. Lo, M., Mamaysky, H., Wang, J.: Foundations of technical analysis: Computational algorithms. Statistical inference and empirical implementation 55(4), 1705–1765 (2000)

13. Michalewicz, Z., Schmidt, M., Michalewicz, M., Chiriac, C.: A decision-support system based on computational intelligence: A case study. IEEE Intelligent Systems 20(4), 44–49 (2005)
14. Mucientes, M., Moreno, L., Bugarin, A., Barro, S.: Evolutionary learning of a fuzzy controller for wall-following behavior in mobile robotics. Soft Comput. 10(10), 881–889 (2006)
15. Oduor, B., Anev, A.: Evolving fuzzy systems. J. Comput. Small Coll. 19(5), 356–357 (2004)
16. Pedrycz, W.: Why triangular membership functions? Fuzzy Sets Syst. 64(1), 21–30 (1994)
17. Ruspini, E.H., Bonissone, P.P., Pedrycz, W. (eds.): Handbook of fuzzy computation (1998)
18. Shi, Y., Eberhart, R., Chen, Y.: Implementation of evolutionary fuzzy systems. IEEE Transactions on Fuzzy Systems 7(2), 109–119 (1999)
19. Surmann, H., Maniadakis, M.: Learning feed-forward and recurrent fuzzy systems: a genetic approach. J. Syst. Archit. 47(7), 649–662 (2001)
20. Wu, C.-J., Liu, G.-Y.: A genetic approach for simultaneous design of membership functions and fuzzy control rules. J. Intell. Robotics Syst. 28(3), 195–211 (2000)

Evolving Regular Expressions for GeneChip Probe Performance Prediction

W.B. Langdon and A.P. Harrison

Departments of Mathematical, Biological Sciences and
Computing and Electronic Systems, University of Essex, UK

Abstract. Affymetrix High Density Oligonuclotide Arrays (HDONA) simultaneously measure expression of thousands of genes using millions of probes. We use correlations between measurements for the same gene across 6685 human tissue samples from NCBI's GEO database to indicated the quality of individual HG-U133A probes. Low concordance indicates a poor probe. Regular expressions can be data mined by a Backus-Naur form (BNF) context-free grammar using strongly typed genetic programming written in `gawk` and using `egrep`. The automatically produced motif is better at predicting poor DNA sequences than an existing human generated RE, suggesting runs of Cytosine and Guanine and mixtures should all be avoided.

1 Introduction

Typically Affymetrix GeneChips (e.g. HG-U133A) measure gene expression at at least eleven points along the gene. Individual measurements are given by short (25 base) DNA sequences, known as probes. These are complementary to corresponding locations in genes. Being complementary, the gene product (messenger RNA) preferentially binds to the probe. Cf. left hand side of Figure 1. Half a million probes are placed on a glass slide in a square grid pattern. A fluorescent dye is used to measure how much mRNA is bound to each probe.

While nothing is simple in Biology, to a first approximation, the amount of mRNA produced by a gene should be the same no matter which part of the mRNA molecule is bound to a probe. Affymetrix groups probes into probesets. Each probeset targets a gene. Therefore probe measurements for the same probeset should be correlated. Figure 1 (right) shows the 110 correlations for a probeset as a "heatmap" (yellow/lighter corresponds to greater consistency between pairs of probes).

There are several biological reasons which might lead to probes on the same gene giving consistently unrelated readings. (Alternative splicing, alternative polyadenylation and 3'-5' degradation, come to mind [11].) However these do not explain all of the many cases of poor correlation. In [9] we found some technological reasons. In particular, [9] showed that probes containing a large ratio of Guanine (G) to Adenosine (A) bases are likely to perform badly. Subsequently we have found that runs of Gs (which will tend to have a high G/A ratio) also

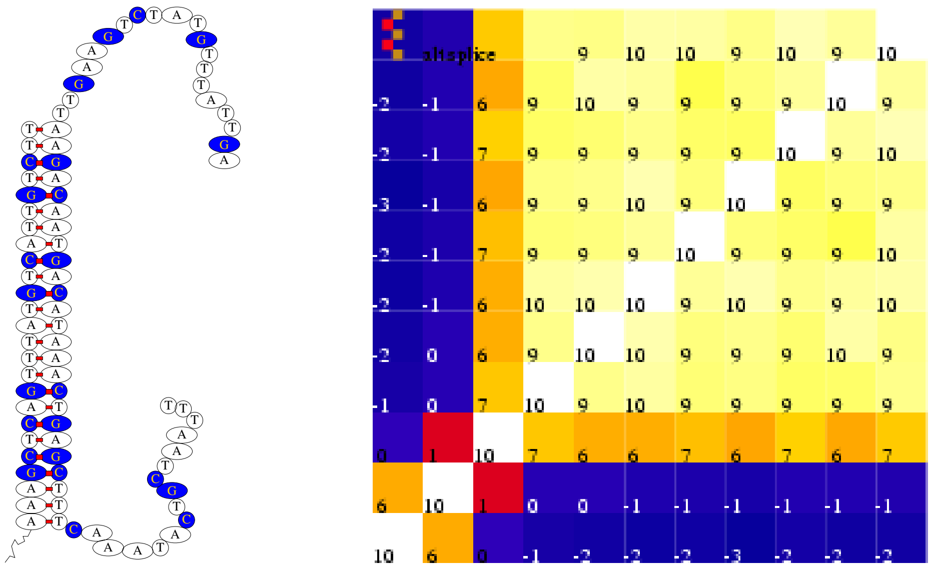


Fig. 1. Left:Schematic of an Affymetrix probe (vertical) bound with complementary target sequence (right). Right: Correlation coefficients ($\times 10$) between 11 probes for gene "S100 calcium binding protein A11" S100A11. Nine of the probes are correlated but PM_1 and PM_2 (bottom 2 rows and 2 left) are not.

tend to indicate problem probes [25]. This has lead us to ask if there are other *sequences* which might indicate dodgy probes.

The next section will briefly describes the research background, whilst Section 3 will describe the preparation of datasets containing the correlation coefficients and probe sequences. Section 4 describes the evolutionary algorithm used to create regular expressions for `egrep`. Section 5 describes how well genetic programming does. Section 6 uses the evolved motif to suggest potential physical explanations for poor probes. Finally, in Section 7 we conclude that several regular expressions, e.g. `G(G|C){4}`, in additions to `GGGG`, can be used together to locate poor probes.

2 Grammars and Evolutionary Computation

Existing research on using grammars to constrain the evolution of programs can be broadly divided in two: Grammatical evolution [21] based largely in Ireland and work in the far east by Wigham [26,27], Wong [28] and McKay [17].

There is quite a body of work on using evolution to induce formal grammars. E.g. Nikolaev tackled the Tomita regular expression benchmarks [20]. GP has also evolved context free grammars [14]. Cetinkaya used grammatical evolution to create regular expression for processing HTML [6].

Ross induced stochastic regular expressions from a number of grammars to classify proteins from their amino acid sequence [24]. Typically his grammars had eight alternatives. In Stockholm regular expressions have been evolved to search for similarities between proteins, again based on their amino acid sequences [7]. Whilst Brameier in Denmark used amino acids sequences to predict the location of proteins by applying a multi-classifier [16] linear GP based approach [4] (although this can be done without a grammar [15]). A similar technique has also been applied to study microRNAs [5].

The next section describes our single stage strongly typed tree based GP being applied to an important DNA problem: explaining why some DNA sequences in wide spread commercial use make poor GeneChip probes.

3 Preparation of Training Data

Previously we had down loaded thousands of experiments from NCBI's GEO [2], normalised them, excluded spatial defects and calculated the correlation between millions of pairs of probes [13,9,12]. To exclude genes which are never expressed, we selected probesets where ten or more non-overlapping probe pairs had correlations of 0.8 or more. For each probe we use the median value of all 10 of its correlations with other members of its probeset (excluding those it overlaps). This gave 4118 probesets, which were evenly split into three to provide independent training, test and validation data.

Previously we found the "mismatch" probes were often poorly correlated with other measurements for the same gene [9]. Since this is known, we excluded them from this study.

As Figure 2 (left) shows correlation coefficients cover a wide range. Since we are using correlation only as an indication of how well a probe is working we decided to exclude the middle values from training and instead use probe pairs that were highly correlated (≥ 0.8) or were very poorly correlated (≤ 0.3). Of the 15 092 available training examples, there are 7 832 probes highly correlated with the rest of their probeset but only 583 poorly correlated. To avoid unbalanced training sets, every generation all 583 negative examples are used and 583 positive examples are randomly chosen from the 7 832 positive examples.

4 Evolving Regular Expression Motifs

4.1 BNF Grammar of Regular Expression

The BNF grammar used (cf. Figure 3) is an extension of that given by Cameron `http://www.cs.sfu.ca/people/Faculty/cameron/Teaching/384/99-3/regexp-plg.html` In particular, matching the beginning of strings (`^`) and the `{n,m}` form of Kleen closure, are also supported. The BNF has been customised for DNA strings. (I.e. `<char>` need only be `A` `C` `G` and `T`). Since various combinations of the start of string symbol, null strings and Kleen closure cause `egrep` to loop, care has been taken to ensure that the new BNF does not permit null strings after `^`.

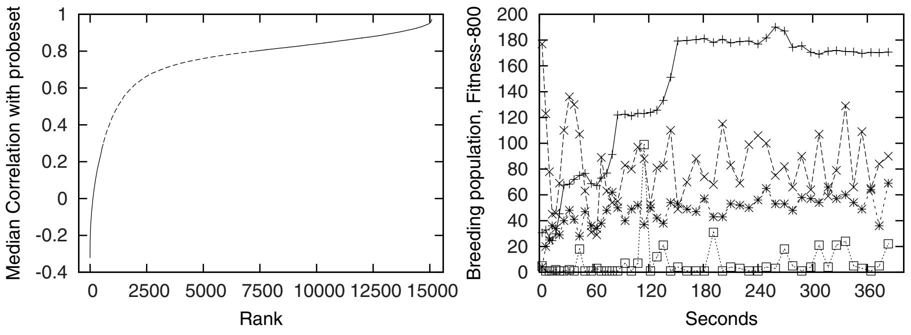


Fig. 2. Training data. Probes with intermediate values (0.3 . . . 0.8) are not used. Right: Evolution of breeding population (best 200 of 1000) of regular expressions to find poor GeneChip probes. Each generation the positive training cases are replaced leading to fluctuations in the measured best fitness (*). However diversity remains high and there are usually few individuals with the same highest fitness (□). In this run the number of distinct phenotypes (×) (i.e. `egrep` search strings) is almost identical to the number of distinct genotypes. Bloat is limited (+), apparently by the tree depth limit (17). However the system slows down by ($\approx \frac{1}{2}$) as evolution proceeds.

Brameier and Wiuf suggests that the traditional `*` and `+` form of Kleen closure are not suitable for bioinformatic applications [5]. Instead they recommend the `{n,m}` form which explicitly defines both lower (`n`) and upper (`m`) limits on the number of times the preceeding symbol must occur. However both `{n,m}` and traditional Kleen closures are used by evolved solutions. To avoid `mutation.awk` seeing "Hamming cliffs", the integer quantifiers used in the `{n,m}` are Gray coded [1]. Similarly the syntax groups together the chemically more similar Pyrimidines (`T` and `C`) and Purines (`A` and `G`).

Our system supports full positive integer values for `minmaxquantifier`, however even modest values can lead `egrep` to hang the computer. Therefore `n` and `m` are limited to 1-9. Finally `egrep` rejects `{n,m}` if `m<n`. This is handled by a semantic rule which removes `,m` from the phenotype when `m` is less than `n`.

4.2 Simplifying the BNF for Use by Evolutionary Algorithms

For simplicity, the BNF is written so that grammar rules are either simple substitution rules (e.g. `<minmaxquantifier>`), rules with exactly two options (e.g. `<RE>`) or terminals (e.g. `"*"` and `T`). In BNF terms, a terminal is a symbol which cannot be substituted in the grammar. Therefore, unlike the BNF rules, it becomes part of the `egrep` regular expression. The simple substitution rules do not have any element of choice. They, like terminals, cannot be chosen as crossover points or targets for mutation. Their principle use is to enable the rules with options to be kept simple.

The binary choice rules are the active parts of the syntax. As they are always binary, each sentence recognised by the BNF has an equivalent binary string.

```
<start>  ::=  <RE>
<RE>  ::=  <union> | <simple-RE>
<union>  ::= <RE> "|" <simple-RE>
<simple-RE>  ::=  <concatenation> | <basic-RE>
<concatenation>  ::= <simple-RE> <basic-RE>
<basic-RE>  ::= <RE-kleen> | <elementary-RE>
<RE-kleen>::= <minmaxquantifier> | <kleen>
<kleen>::= <star> | <plus>
<star>  ::= <elementary-RE2> "*"
<plus>  ::= <elementary-RE2> "+"
<minmaxquantifier>   ::= <elementary-RE4> "{" <int> <optREint> "}"
<elementary-RE>  ::= <group> | <elementary-RE1>
<elementary-RE1>  ::= <xos> | <elementary-RE2>
<elementary-RE2>  ::= <any> | <elementary-RE3>
<elementary-RE3>::= <set> | <char>
<elementary-RE4>  ::= <group> | <elementary-RE2>
<group>  ::=  "(" <RE> ")"
<xos>  ::=  <sos> | "$"
<sos>  ::=  "^" <elementary-RE4>
<set>  ::=  <positive-set> | <negative-set>
<positive-set>  ::=  "[" <set-items> "]"
<negative-set>  ::=  "[^" <set-items> "]"
<set-items>  ::=  <set-item> | <set-items2>
<set-items2>  ::=  <set-item> <set-items>
<set-item>  ::=  <char>
<char>  ::=  <c00> | <c01>
<any>  ::=  "."
<c00>  ::=  T | C
<c01>  ::=  A | G

<optREint>  ::=  <2ndint> | $
<2ndint>  ::=  "," <int>
<int>  ::=  <d0>
#4 Bit Gray Code Encoder
<REdigit>  ::=  <d111> | <d0>
<d0>  ::=  <d00> | <d01>
<d00>  ::=  <d000> | <d001>
<d01>  ::=  <d010> | <d011>
<d000>  ::=  1
<d001>  ::=  3 | 2
<d010>  ::=  7 | 6
<d011>  ::=  4 | 5
<d111>  ::=  8 | 9
```

Fig. 3. Grammar used to specify legal regular expressions for use as `egrep` search strings for testing DNA sequences

(A BNF sentence means an `egrep` regular expression in our case.) Each bit corresponds to a BNF rule with two options needed when parsing the sentence. The bit indicates which option should be invoked. Note that the meaning of each bit depends upon where we have reached in parsing the sentence (i.e. on the earlier bits). In traditional GP terms, this list of choices is the evolving tree. The BNF grammar acts to put labels on the choices, which constrain crossover, but it is the choices (not the BNF grammar) which is the genetic material of the evolving individual. Note, unlike grammatical evolution, strongly typed crossover respects [23] the meaning of the BNF rules.

4.3 Creating Random BNF Sentences

The initial random population is created using ramped half-and-half [8]. It may help to think of this as applying the usual genetic programming ramped half-and-half algorithm to a binary tree (of choice nodes).

We start from `<start>` and recursively follow the BNF. However when we reach a rule with options we need to choose one. As in ramped half-and-half we keep track of how deep we are nested. If we have not reached the depth needed to terminate the recursion, we randomly choose one of the options. This is the equivalent of choosing a GP function. (As with other strongly typed GPs, if a chosen route through the syntax has no further choices to be made, we may be forced to terminate a recursive branch early.)

To terminate a recursion we choose the "simpler" option. Our BNF has been written so that the simpler option is always on the right. (This is flagged by `RE` in the rule name.) If there is no "simpler" choice, the choice is made randomly. This mechanism is also used for mutating existing regular expressions.

Although this may seem complex, `gawk` (Unix' free interpreted pattern scanning and processing language) can handle populations of a million individuals.

4.4 Crossing over BNF Sentences

Creating a new sentence from two high fitness sentences is essentially subtree crossover [22] applied to the binary choice tree (cf. Section 4.1) with the addition of strong type constraints [18]. This is implemented by scanning the grammar used to create the first parent for all the rules with two options. One of these is randomly chosen. For example, suppose the first parent starts `<start> <RE> <union>` and suppose `<union>` is chosen as the crossover point. For a grammatically correct child to be produced all that is necessary is that the crossover point chosen in the second parent should also be `<union>`. (There are complications to do with depth and size limits, which we shall ignore for the time being.) Therefore the second parent is scanned to find all occurrences of `<union>`. One of them is randomly chosen to be the second crossover point. (If there are none, this crossover is aborted and another initial crossover point is chosen. If we keep failing, eventually another pair of parents is chosen.)

Crossover is based on normal GP subtree crossover, cf. [22, Figure 2.5]. The new child is created by copying the start of the first parent, excluding the subtree at the first parent's crossover point. Then genetic material from the subtree at the second parent's crossover point is added. Finally the remainder of the first parent is appended to the child. This is implemented by crossing over the binary choice trees to create a binary choice tree for the new child. Apart from issues

Table 1. Strongly Typed Grammar GP for GeneChip Correlation Prediction

Primitives:	The function and terminal sets are defined by the BNF grammar (cf. Figure 3). BNF rules with two options correspond to binary GP functions. The rest of the BNF grammar correspond to GP terminals.
Fitness:	true positives+true negatives. (I.e. proportional to the area under the ROC curve or Wilcox statistic [10].) Less large penalty if `egrep` fails or it matches all probes or none.
Selection:	(200,1000)
Initial pop:	Ramped half-and-half 3:7
Parameters:	100% subtree crossover. Max tree depth 17 (no tree size limit)
Termination:	50 generations

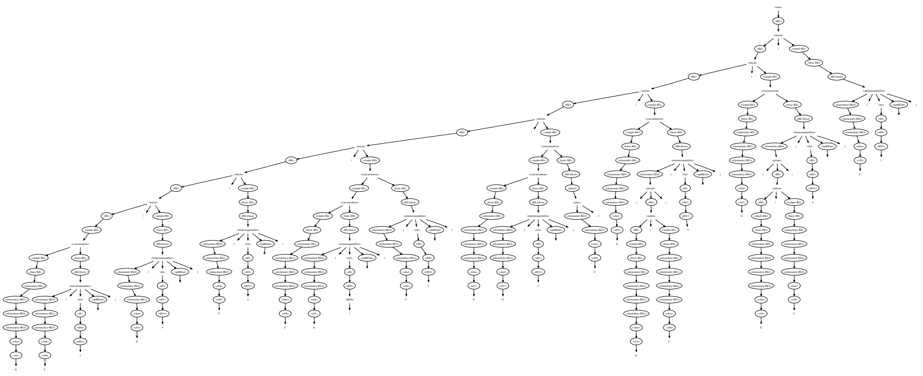


Fig. 4. Geneotype of best program in generation 50. Active choice nodes in the BNF (cf. Figure 3) are emphasised by placing them in ovals. The resulting phenotype is simply the 58 leaf nodes (excluding $), read in left to right order: `GC{3}|G{4}|C{4}|CG{1}C{2}|GG{4}C+|G(G|C){4}|G(G|C)4|C{3}`. It is equivalent to `GGGG|CGCC|G(G|C){4}|CCC`.

of tree size and depth, we are guaranteed that the new binary choice tree will represent a valid sentence in the BNF grammar.

The final step is to recursively trace through the BNF grammar, obeying the binary choices. Each time an BNF terminal (except the null symbol) is encountered it is appended to the new regular expression. In order to be able to breed following generations, the new individual's genotype must also be passed to the next generation. For convenience, this is done by writing each BNF symbol, as it is encountered, to the file holding the next generation.

4.5 Evaluating the Fitness of BNF Sentences

Each generation, a command file is generated which contains a `egrep -c -v '`*RE*`'` command for each individual in the population. (*RE* is the individual's regular expression.) The command is run on a file holding the DNA sequences of the 583 probes poorly correlated with the rest of their probeset. The same command is also

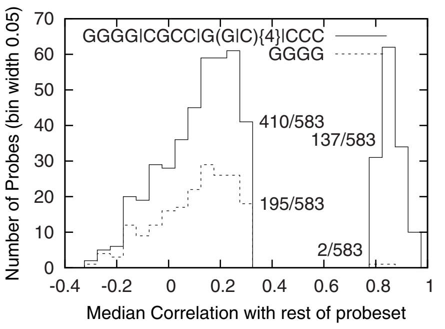


Fig. 5. Performance of evolved motif on its training data versus human generated motif (dashed)

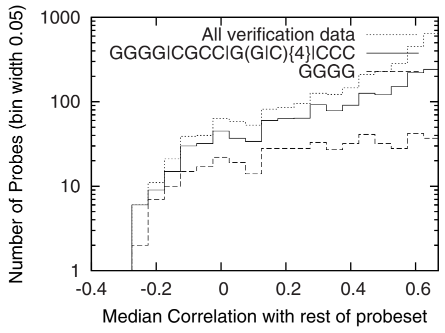


Fig. 6. Performance of evolved and human generated motifs on verification data

run on a file holding the 583 positive probes selected for use in this generation. The fitness of the regular expression is based on the difference between the number of lines in the two files which match *RE*. Expressions which either match all probes or fail to match any are penalised by subtracting 583 from their score. See also Table 1. Implementation details can be found in [12].

5 Results

By the end of the first run, with a population of 1000 (cf. Table 1 and Figure 2, right) GP had evolved a probe performance predictor (see Figure 4) equivalent to `GGGG|CGCC|G(G|C){4}|CCC`. It is obvious that it includes the previous rule (`GGGG`, [25]) but includes other possibilities. Therefore it finds more poor probes. Inevitably it will also incorrectly predict more high correlation probes are poor but the increase is more than offset by its better performance on the poor probes. See Figure 5. It has a fitness of 856. (`GGGG` has a fitness of 776.)

The confusion matrix for the evolved regular expression on the whole of the training set (including the 6677 positive middling values which GP never saw) is $\boxed{\begin{array}{rr}410 & 4448\\ 173 & 10061\end{array}}$ and on the verification data: $\boxed{\begin{array}{rr}448 & 4436\\ 174 & 10045\end{array}}$. (The corresponding matrices for `GGGG` are $\boxed{\begin{array}{rr}195 & 479\\ 388 & 14030\end{array}}$ and $\boxed{\begin{array}{rr}209 & 434\\ 413 & 14047\end{array}}$.) Unlike in many machine learning applications, there is no evidence of over fitting. Indeed the corresponding results for the test set (second matrix of each pair) are not significantly different (χ^2, 3 dof) from those on the whole training set. The evolved regular expression picks up significantly more (χ^2, 3 dof) (448 v. 209) poorly performing probes on the validation set than the human produced regular expression. Figure 6 shows the number of DNA probes matching the evolved motif against their average correlation with the rest of their probeset.

As is common in optimisation [3], almost all the time is taken by the fitness evaluation. In our case, elapse time is dominated by the command script which runs `egrep -c`. Typically this takes 8.5mS per individual. The time taken by `gawk` to process the BNF grammar, perform crossover, generate the regular expressions, etc., is negligible.

6 Discussion

Theoretical and empirical studies of GeneChips confirm that the behaviour of DNA probes tethered to a glass surface can be quite different from DNA behaviour in bulk solution. This is a new and difficult area and there are not deep pure Physics experimental results. Therefore experimental studies have concentrated on data gathered during normal operation of the chips.

Our automatically generated motif, suggests that in addition to Gs, Cs are important. Indeed the fact that only three consecutive Cs is predictive (whereas four Gs are needed) suggests that Cs are more important than Gs. It is known in GeneChips DNA C–G RNA binds more strongly than DNA G–C RNA [19]. (Both C and G bind more strongly than A and T.) We are tempted to suggest that a CCC sequence on a DNA probe can act as a nucleation site encouraging the probe to bind to GGG on RNA. Indeed the evolved motif suggests that four Gs and mixtures of five Cs and Gs might also form nucleation sites.

The sequence CCC is too short to be specific to a particular gene. GeneChips are designed on the assumption that only RNA sequences which are complementary to the full length of the probe will be stable. However studies have shown that nonspecific targets can be bound to GeneChip probes for several hours even if held only by the nucleation site. This may be why probes with quite short runs of either Cs or Gs can be poorly correlated with others designed to measure the same gene.

7 Conclusions

Access to the raw results of thousands of GeneChips (each of which costs several hundreds of pounds) makes new forms of bioinformatic data mining possible.

Millions of correlations between probes in the same probeset, which should be measuring the same gene, show wide variation. Evolution of regular expressions confirms previous work [9,25] that the DNA sequences from which the probes themselves are formed can indicate poor probe performance. Indeed several new motifs (e.g. CCC) which predict probe quality have been automatically found.

Linux code is available via FTP `ftp://cs.ucl.ac.uk/genetic/gp-code/`

References

1. Thomas B.: Evolutionary Algorithms in Theory and Practice. OUP (1996)
2. Barrett, T., et al.: NCBI GEO: mining tens of millions of expression profiles–database and tools update. Nucleic Acids Research 35, D760–D765 (2007)
3. Beyer, H.-G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001)
4. Brameier, M., Krings, A., MacCallum, R.M.: NucPred predicting nuclear localization of proteins. Bioinformatics 23(9), 1159–1160 (2007)
5. Brameier, M., Wiufp, C.: Ab initio identification of human microRNAs based on structure motifs. BMC Bioinformatics 8, 478 (2007)

6. Cetinkaya, A.: Regular expression generation through grammatical evolution. In: Yu, T. (ed.) GECCO-2007 workshop program, pp. 2643–2646. ACM Press, New York (2007)
7. Handstad, T., Hestnes, A.J.H., Saetrom, P.: Motif kernel generated by GP improves remote homology and fold detection. BMC Bioinformatics 8(23)
8. Koza, J.R.: Genetic Programming. MIT press, Cambridge (1992)
9. Langdon, W.B.: Evolving GeneChip correlation predictors on parallel graphics hardware. In: WCCI, Hong Kong, June 1-6, 2008, pp. 4152–4157. IEEE, Los Alamitos (2008)
10. Langdon, W.B., Barrett, S.J.: GP in data mining for drug discovery. In: Ghosh, A., et al. (eds.) Evolutionary Computing in Data Mining, pp. 211–235 (2004)
11. Langdon, W.B., da Silva Camargo, R., Harrison, A.P.: Spatial defects in 5896 HG-U133A GeneChips. In: Dopazo, J., et al. (eds.) CAMDA 2007 (2007)
12. Langdon, W.B., Harrison, A.P.: A grammar based strongly typed genetic programming system for finding regular expression which predict affymetrix DNA probe performance. Technical report, CES-483, University of Essex, UK (2008)
13. Langdon, W.B., Upton, G.J.G., da Silva Camargo, R., Harrison, A.P.: A survey of spatial defects in Homo Sapiens Affymetrix GeneChips (submitted)
14. Langdon, W.B.: Genetic Programming and Data Structures. Kluwer, Dordrecht (1998)
15. Langdon, W.B., Banzhaf, W.: Repeated sequences in linear genetic programming genomes. Complex Systems 15(4), 285–306 (2005)
16. Langdon, W.B., Buxton, B.F.: Evolving receiver operating characteristics for data fusion. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tetamanzi, A.G.B., Langdon, W.B. (eds.) EuroGP 2001. LNCS, vol. 2038, pp. 87–96. Springer, Heidelberg (2001)
17. McKay, R.I., Hoang, T.H., Essam, D.L., Nguyen, X.H.: Developmental evaluation in GP. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 280–289. Springer, Heidelberg (2006)
18. Montana, D.J.: Strongly typed GP. Evolutionary Computation 3(2), 199–230
19. Naef, F., Wijnen, H., Magnasco, M.: Reply to comment on solving the riddle of the bright mismatches. Physical Review E 73(6), 063902 (2006)
20. Nikolaev, N.I., Slavov, V.: Concepts of inductive genetic programming. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) EuroGP 1998. LNCS, vol. 1391, pp. 49–60. Springer, Heidelberg (1998)
21. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE TEC 5(4), 349–358 (2001)
22. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming (With contributions by J. R. Koza) (2008), http://www.gp-field-guide.org.uk
23. Radcliff, N.J.: Genetic set recombination. In: FOGA 2, pp. 203–219. Morgan Kaufmann, San Francisco
24. Ross, B.J.: The evaluation of a stochastic regular motif language for protein sequences. In: Spector, L., et al. (eds.) GECCO 2001, pp. 120–128 (2001)
25. Upton, G.J., Langdon, W.B., Harrison, A.P.: Incorrect measurement of gene expression by microarrays (submitted)
26. Whigham, P.A.: Search bias, language bias, and genetic programming. In: Koza, J.R., et al. (eds.) Genetic Programming 1996, pp. 230–237. MIT Press, Cambridge (1996)
27. Whigham, P.A., Crapper, P.F.: Time series modelling using GP: In rainfall-runoff models. In: Spector, L., et al. (eds.) AiGP3, pp. 89–104. MIT Press, Cambridge (1999)
28. Wong, M.L., Leung, K.S.: Evolving recursive functions for the even-parity problem using genetic programming. In: AiGP 2, pp. 221–240. MIT Press, Cambridge (1996)

Evolutionary Market Agents for Resource Allocation in Decentralised Systems

Peter R. Lewis[1], Paul Marrow[1,2], and Xin Yao[1]

[1] University of Birmingham, Birmingham, UK
[2] BT Group plc, Adastral Park, Ipswich, UK
p.r.lewis@cs.bham.ac.uk, paul.marrow@bt.com, x.yao@cs.bham.ac.uk

Abstract. We introduce self-interested evolutionary market agents, which act on behalf of service providers in a large decentralised system, to adaptively price their resources over time. Our agents competitively co-evolve in the live market, driving it towards the Bertrand equilibrium, the non-cooperative Nash equilibrium, at which all sellers charge their reserve price and share the market equally. We demonstrate that this outcome results in even load-balancing between the service providers.

Our contribution in this paper is twofold; the use of on-line competitive co-evolution of self-interested service providers to drive a decentralised market towards equilibrium, and a demonstration that load-balancing behaviour emerges under the assumptions we describe.

Unlike previous studies on this topic, all our agents are entirely self-interested; no cooperation is assumed. This makes our problem a non-trivial and more realistic one.

Keywords: decentralised systems, market-based control, co-evolution, load-balancing, self-interested agents.

1 Introduction

Emerging paradigms for the development and deployment of massively distributed computational systems allow resources to span many locations, organisations and platforms, connected through the Internet [1]. It has been predicted that the majority of transactions over the Internet will, in the future, be carried out by autonomous agents on behalf of their owners [2]. In this scenario, neither control nor even full knowledge of key resources may be assumed. There is therefore a need to find novel ways to understand and autonomically manage and control these large, decentralised and dynamic systems [3].

Gupta et al. [4] propose externality pricing for the provision of otherwise virtually zero cost per-use computational services. They argue that this approach, where service users self-select their quantity based on price, is a more preferable approach to the alternative of provider-side enforced quantity limits.

We propose the use of autonomous *evolutionary market agents* as an approach to achieving this. Evolutionary market agents operate on behalf of individual actors in a decentralised market-based system. Such decentralised markets have no auctioneers or market-makers. Instead, selling agents (the service providers) advertise their services at a price, and buying agents (the service users) decide whether to buy, and from whom. Making use of only local information about the live market, the sellers adapt (or evolve) the offers of their host in order to maximise their payoff. When coupled with rational, self-interested buying agents, we demonstrate that this approach is able to provide load-balanced allocations across the system as a whole.

Market-based resource allocation and adaptive pricing are not new ideas (see [5] for an introduction) and our contribution in this paper is twofold. Firstly, we describe how decentralised computational markets can provide an emergent load-balancing behaviour between self-interested agents at equilibrium, and secondly, we demonstrate the use of competitive co-evolution to drive the market towards this equilibrium.

2 Related Work

Traditionally, load-balancing is done in a centralised manner [6]. Relying on a single node, such approaches have a central point of failure. Alfano and Di Caprio note that scalability is a critical factor in load-balancing systems [7]. They present a scalable, decentralised load-balancing mechanism, based upon cooperating peers. Our model, however, does not require cooperation. Indeed, its power lies in the self-interested competition of peers, over whom we may not have control.

As such, our approach falls into the broad category of market-based control, a methodology which has been applied to the allocation of resources in various real-world scenarios. Clearwater provides a useful introduction to the use of computational markets in scenarios such as bandwidth allocation and air-conditioning control [5]. A full review is beyond the scope of this paper, but may be found in [8].

Cliff and Bruten note that a large proportion of market-based control systems, however, either rely on a central auctioneer, or require human intervention [8]. Therefore, though much of the computation is done by individual agents and is distributed, these systems are often not decentralised. They argue that this leads to a brittleness of the system.

A number of distributed auction mechanisms have also been proposed [9,10,11,12], which do not rely on one central auctioneer. These reduce the fragility associated with reliance upon a single point, provide more scalability and allow for dynamic composition of auctions. Typically, the central auctioneer is replaced by a number of local ones, which communicate through some secure means. However, similarly to the vulnerabilities of the Internet's domain name system [13], failure at certain points in the network may well cripple wider functionality, at best.

Kuwabara et al. propose what we believe to be the most decentralised market-based approach to the allocation of resources [14]. Here, no auctioneer, specialist or market-maker is used; prices are set solely by the sellers and advertised via a broadcast mechanism. Rational buyers then decide the quantity to purchase from each seller, in order to maximise their payoff. However, unlike our sellers' strategies, those used used by Kuwabara et al. are not driven by self-interest. The problem studied here is of the same general form as that in [14].

A variety of computational intelligence techniques have been used to attempt to replicate the behaviour of human marketing managers in the real world. Midgley et al. model brand managers in the retail coffee market as agents, which determine the occurrence and nature of various promotions and price discounts [15]. However, the difference between these cases and our problem is that we are not interested in modelling human behaviours, or in replicating them.

Applying evolutionary techniques to self-interested selling behaviour, Cheung et al. [16] used a simple model of the Australian consumer petroleum market to predict how sellers would modify their prices in a competitive environment. By giving sellers historical information about each other, they correctly replicated *implicit cartel* behaviour found in the real world, where despite the short-term rationality of cutting prices to increase market share, sellers in fact raised their prices in step. Their simulations resulted in sellers colluding in order to all charge the maximum price and share the market equally. If all other sellers cooperate, then this is the optimal strategy from a seller's point of view. However, only one seller not cooperating and undercutting the others, leads to it making a greater short-term payoff.

3 Problem Formulation

We consider a scenario consisting of a set of service providing nodes, S, each member of which provides an equivalent, quantitatively divisible service, the resource π, which may vary only in price. We assume that each member of S has an equal capacity for the provision of π, and that they cannot be relied upon to cooperate. We then imagine a large population of service users or buyers, B, each member of which aims to consume some of the resource π, at regular intervals. Our objective is to balance the load, such that all the service providers in S are providing an equal amount of π across the population of service users. Though this is a trivial problem when cooperation may be assumed, we wish to achieve this using self-interest, with no central control. Note that we are not modelling service providing nodes owned by competing businesses in the real-world, since then load-balancing would not be desirable; self-interested competition is instead artificially created in order to serve the purposes of the system owner.

At a given instant, a service provider, $s_i \in S$, advertises π at the price $p^{\pi}_{s_i}$ per unit. From a service user's point of view, this may be denoted as the offer X_{s_i}. Each service user, a buyer in this case, purchases some of the resource π should it be in their interest to do so at the price offered. The system iterates, with service providers able to adapt their prices to the market conditions over time. The actual provision of π may be regarded as instantaneous, such that it does not interfere with this mechanism.

For simplicity at this stage, we assume that the system proceeds in discrete time-steps, that each buyer $b_j \in B$ desires exactly one unit of π per time-step, and that each and every $s_i \in S$ has sufficient quantity of π available to satisfy all the buyers in B should it be so requested. This final assumption is commonplace in the provision of information-based services, and is present in other related work such as [14]. We do not believe that its presence alters the underlying behaviour we demonstrate. We further simplify the model by stipulating that each service provider has no overhead cost associated with obtaining, producing or providing π. This assumption allows for buyers to purchase a tiny amount from each of a large number of sellers. Whether or not this is unrealistic will be determined by the application scenario.

Each time-step, each buyer, if it chooses to buy, may purchase any amount of π from any number of service providers in S, subject to the constraint that the total amount purchased per time-step is equal to exactly one unit. If no offer from any $s_i \in S$ is in its interest, the buyer may instead opt to purchase nothing. We therefore define q_{ij} to be the instantaneous quantity bought by buyer b_j from seller s_i. The constraints mean therefore that $\sum_{i=1}^{|S|} q_{ij} \in \{0, 1\}$ for all $b_j \in B$.

The quantity of π sold by a given seller s_i at a given time-step, its *load*, l_{s_i}, is therefore:

$$l_{s_i} = \sum_{j=1}^{|B|} q_{ij} \,. \tag{1}$$

Our stated objective is even load-balancing. In previous work [14], this is defined as being the minimisation of the variance between the service providers' loads, v_l, as shown in equation 2.

$$v_l = \frac{\sum_{i=1}^{|S|} (l_{s_i} - \mu)^2}{|S|} \,, \tag{2}$$

where

$$\mu = \frac{\sum_{i=1}^{|S|} l_{s_i}}{|S|} \,.$$

However, we prefer instead the measure, d_l, a normalised measure of mean absolute distance from the ideal load (here referred to as *NMA distance*), as described in equation 3. We find that this scales better with respect to $|S|$, making comparisons between simulations with different numbers of sellers simpler. Hence a high d_l indicates an uneven load, while a perfectly even load leads to a value of zero. We define μ as before.

$$d_l = \frac{\sum_{i=1}^{|S|} |l_{s_i} - \mu|}{|S|} \div \frac{2|S| - 2}{|S|^2} \,. \tag{3}$$

Both buyers and sellers accrue a payoff, or utility gain, from their interactions in the marketplace. For buyers, this will be the value they associate with the price paid subtracted from the value they associate with the purchased resource. We assume that buyers are self-interested, such that they attempt to maximise their utility. However, as in previous work [14], we do not assume that they are hyperrational, behaving in an all-or-nothing manner in favour of the instantaneously most attractive option, since this behaviour may expose the buyer to a degree of risk.

Instead, we investigate buyers who are *risk-averse*, preferring to spread their purchases across a number of sellers. At each time-step, each buyer looks at the available offers, $X_{s_i} \forall s_i \in S$, and purchases a proportion of their total desired resource from each seller, relative to the expected utility gain from selecting the offer from that seller. An alternative would be to motivate risk-aversion through the game itself, however we prefer not to complicate the model, instead favouring the clarity gained by the assumption of risk averse behaviour.

Therefore firstly a buyer considers a unit transaction of π from each seller. The instantaneous expected utility, or payoff, for a buyer, b_j, in purchasing one unit of π from the seller s_i at the current offer,

$$E[U^{b_j}(X_{s_i})] = u^{b_j}(p^{\pi}_{s_i}, \pi) , \tag{4}$$

where $E[U^{b_j}(X_{s_i})]$ represents buyer b_j's expected payoff from accepting offer X_{s_i}, and $u^{b_j}(p, \pi)$ is the buyer's utility function over the goods: money and π.

The buyer then purchases a proportion of his total desired π from each seller. Any sellers which would provide a negative payoff are ignored. Recalling that q_{ij} is the quantity of π purchased by buyer b_j from seller s_i,

$$q_{ij} = \frac{E[U^{b_j}(X_{s_i})]}{\sum E[U^{b_j}(X_{s_k})]} , \tag{5}$$

which ranges over k for which the expectation is positive.

The market model we have described here is an example of a Bertrand Game [17]. This is where two or more sellers compete by simultaneously setting prices for equivalent goods, and buyers then decide the quantity to purchase from each seller in a rational utility-maximising manner. The theoretical non-cooperative Nash equilibrium outcome is the Bertrand equilibrium, at which all sellers charge their reserve price and share the market equally. It is the equal sharing of the market at the Bertrand equilibrium which provides us with load-balancing.

However, Bertrand competition relies on the presence of a number of potentially unrealistic assumptions. Two of these are of particular interest. Firstly, Bertrand competition assumes no collusion between sellers. Cheung et al. showed that if sellers are able to reliably predict the behaviour of competitors, then they may implicitly collude in order to raise prices and hence their payoffs [16]. This behaviour is not observed in our model however, since the sellers do not retain historical information concerning each other. Such an ability might well be unfeasible in very large systems [16], though clearly a heterogeneous set of sellers, differentiated by strategic ability is likely to lead to an uneven market, and hence an uneven resource allocation.

Secondly, Bertrand competition assumes that sellers compete only on price, and are otherwise unable to differentiate their products in the market. This is unlikely, since more realistic service providing nodes' differing quality of service will provide product differentiation. Once competition exists other than on price, Bertrand competition no longer applies, though other potentially useful equilibria will exist.

However, the introduction of issues other than price, and with it the heterogeneity of buyers, will also introduce the likelihood of 'price-war' behaviour. Kephart et al. showed that in certain circumstances the competitive behaviour of (non-colluding) sellers would lead to a never-ending series of price-wars [18]. This is shown to be the case when services are described over a number of issues, and a heterogeneous population of buyers have preferences over those issues such as to exist in different *niches* of the market. In these circumstances, sellers undercut each other in order to gain a greater market share and hence greater payoff. However, once the price has become sufficiently low, it becomes rational for the sellers to switch to providing for another niche of buyers, and the competition begins again. Once competition in that niche has driven the price down, the sellers will switch to a different niche, and so on. This result will be important to consider in future work.

4 Evolutionary Market Agents

Sellers also accrue a payoff, their *revenue*. In our model, this is income from the sale of π. At a given instant, the revenue of a seller, $s_i \in S$, is therefore:

$$r_{s_i} = \sum_{j=1}^{|B|} p^{\pi}_{s_i} q_{ij} \,. \tag{6}$$

Ideally, a seller will wish to maximise its revenue by increasing both its price and its market share, however as we have seen, the market share will depend upon the relationship between its price and those of its competitors.

An evolutionary market agent operates on behalf of each seller, with the self-interested objective of maximising its revenue. Using evolutionary computation techniques, the agent evolves the market position of its host over time. In this model, a market position consists simply of price, therefore each individual represents a real-valued price. For each interaction in the market, the price encoded by an individual is adopted, and the resulting payoff provides its fitness.

The evolutionary algorithm for seller s_i's agent proceeds as follows:

1. Decide upon the design parameters to be used: initial price range, population size and mutation factor. In the simulations described, an initial price range of 0 to 500 was chosen, along with a population size of 10 and mutation factor (α) of 0.1.
2. Generate an initial population, P, and set $k = 1$. Each individual in P is a real value, drawn from the uniform random distribution $[0, p_{max}]$.
3. *Initial fitness testing*
   (a) Set the seller's offer to the value of the first individual in P, and enter the market for one market time-step. Record the seller's revenue, r_{s_i} as that individual's fitness.
   (b) Repeat for the next individual in P, until all initial individuals have been fitness tested in the market.

4. *Probabilistic tournament selection*
   (a) Select four individuals, x_1, x_2, x_3 and x_4 from P, at random, such that $x_1 \neq x_2 \neq x_3 \neq x_4$.
   (b) Let champion c_1 be either x_1 or x_2, the fitness of whichever is greater with probability 0.9, the fitness of whichever is less otherwise.
   (c) Let champion c_2 be either x_3 or x_4, the fitness of whichever is greater with probability 0.9, the fitness of whichever is less otherwise.
5. Let the offspring, o, be a new individual with its value equal to the mid-point of the values of c_1 and c_2.
6. Mutate o, by perturbing its value by a random number drawn from a normal distribution with mean zero and standard deviation α.
7. Select the individual in $\{x_1, x_2, x_3, x_4\}$ with the lowest fitness value, remove it from P, and insert o into P.
8. Set the seller's offer to the value encoded in o, and enter the market for one market time-step. Record the seller's revenue, r_{s_i} as o's fitness.
9. Repeat from step 4.

5 Simulation Results

5.1 A Baseline Scenario

We firstly consider a small scenario with two service providers, such that $S = \{s_1, s_2\}$, each providing the resource π, at prices $p^{\pi}_{s_1}$ and $p^{\pi}_{s_2}$ respectively. Both s_1 and s_2 make use of an evolutionary market agent, as described in section 4 in order to determine these prices at each time-step.

We begin with 10 buyers (the service users), with identical linear utility functions,

$$u^{b_j}(p^{\pi}_{s_i}, \pi) = 375 - p^{\pi}_{s_i} \, . \tag{7}$$

This represents a buyer being indifferent between a unit of π and its cost at a price of 375. This is an arbitrary positive value, and has little impact other than to provide a range of positive payoff values for a range of prices. An alternative approach would be to have the objective of minimising the buyer's spending, though this would remove the notion of a value placed upon π by the buyer. Actively considering negative buyer payoffs would introduce the question of buyer motivation, and considering a payoff able to range over both positive and negative values would slightly complicate the buyer's decision function for no gain. Linear utility functions are used to give an estimation of a service user's expected preferences, though the exact function will of course depend on the specific service and its users. The success of our approach with other forms of utility function remains a topic for research.

Figure 1 shows the normalised mean absolute distance from the ideal load, d_l between s_1 and s_2, over time, for a typical run and across 30 independent runs.

Here, the NMA distance drops quickly, indicating the approach's ability to achieve a roughly even load between the two service providers in a short time.

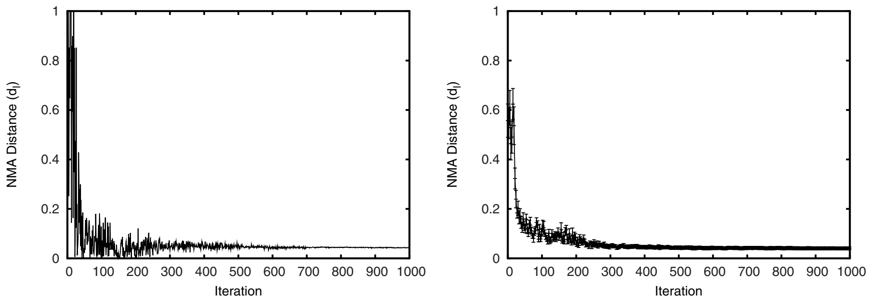


Fig. 1. Load balancing performance over time: 2 sellers and 10 buyers. Typical run (left) and mean and standard deviation over 30 independent runs (right).

This this is due to the evolutionary agents' competitively co-evolving their prices to within close proximity of each other, resulting in roughly even shares of the market. Due to diverse populations within each agent however, their prices, and hence the allocation of resources, continue to vary.

Following these exploratory fluctuations, the NMA distance then stabilises as the populations converge. At this point, the load-balance is highly equal. It is important to note that Kephart's price-wars [18], which would result in an unstable load allocation, do not occur here. This is the case since the sellers describe the service π over only a single issue, price, and hence our buyer population is homogeneous.

5.2 A More Complex Scenario

Due to the distributed, decentralised nature of our approach, it is highly scalable. The complexity of the agents' evolutionary algorithms remains constant with respect to the number of buyers, whilst the complexity of the buyers' algorithm grows only linearly with respect to the number of sellers.

Figure 2 shows both a typical run and mean and standard deviation calculated over 30 independent runs for $|S| = 1000, |B| = 10,000$.

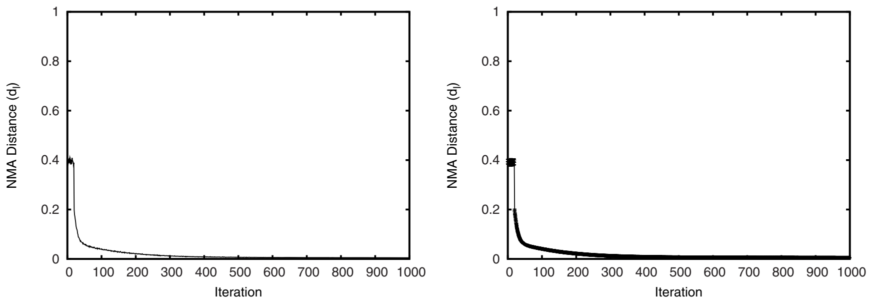


Fig. 2. Load balancing performance over time: 1,000 sellers and 10,000 buyers. Typical run (left) and mean and standard deviation over 30 independent runs (right).

Here results are of a similar form to the previous simulation. This shows the power of our approach to achieve load-balancing in a large, decentralised system where no individual has any desire in favour of this behaviour. Indeed to the contrary, though our outcome is similar to that in [14], our approach is novel in that it relies on self-interested behaviour. The presence of larger populations of agents appears to lead to more reliable results; the approach clearly scales well.

6 Conclusions and Future Work

We have presented a resource allocation problem, motivated by an emerging computational paradigm; dynamic, decentralised, service-based systems. Based on the mechanism proposed by Kuwabara et al. [14], we have described a decentralised, evolutionary market-based approach, which makes use of Bertrand competition between self-interested sellers to achieve load-balancing. No cooperation is assumed. We believe that our approach is more suited to this scenario than other decentralised load-balancing mechanisms, since it accounts for self-interested utility maximising individuals.

Unlike the majority of market-based systems, our approach requires no central point of control or auctioneer. Agents have no knowledge of the size of the marketplace or any history. It has no point which is weaker than any other, and is hence both scalable and robust to failure. Sellers instead have the ability to advertise their prices through a broadcast mechanism.

A future, more realistic model may include a second issue with which to describe quality of service. Sellers could then achieve product differentiation. Outcomes in such a scenario are likely to be more complex than in the model investigated here, in which effectively sellers behave as Dutch auctioneers, since in general their prices only reduce over time.

It is also likely that more realistic scenarios will be dynamic, where service providers may be added to or removed from the system. In addition, the population of buyers may change over time, and there may also be external disturbances. It would be desirable for the system to autonomically achieve a new load-balance in the presence of such changes.

Finally, more advanced tuning of the evolutionary algorithm used in the sellers' evolutionary market agents should improve system performance, and analysis of the algorithm's properties, especially in dynamic environments, will be useful in achieving this. Adaptive mutation, in order to allow sellers to explore the market widely when necessary, but to compete without reckless price changes, will be one potentially useful technique.

References

1. Singh, M.P., Huhns, M.N.: Service-Oriented Computing: Semantics, Processes, Agents. John Wiley and Sons, Chichester (2005)
2. He, M., Jennings, N.R., Leung, H.: On agent-mediated electronic commerce. IEEE Transactions on Knowledge and Data Engineering 15(4), 985–1003 (2003)

3. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: Research roadmap (2006)
4. Gupta, A., Stahl, D.O., Whinston, A.B.: The economics of network management. Communications of the ACM 42(9), 57–63 (1999)
5. Clearwater, S.H. (ed.): Market-Based Control: A Paradigm for Distributed Resource Allocation. World Scientific, Singapore (1996)
6. Ardellini, V.C., Olajanni, M.C., Yu, P.S.: Dynamic load balancing on web server systems. IEEE Internet Computing 3(3), 28–39 (1999)
7. Alfano, R., Caprio, G.D.: Turbo: an autonomous execution environment with scalability and load balancing features. In: Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and its Applications (DIS 2006), pp. 377–382 (2006)
8. Cliff, D., Bruten, J.: Simple bargaining agents for decentralized market-based control. Technical Report HPL-98-17, HP Laboratories, Bristol, UK (1998)
9. Esteva, M., Padget, J.: Auctions without auctioneers: Distributed auction protocols. In: Moukas, A., Ygge, F., Sierra, C. (eds.) Agent Mediated Electronic Commerce II. LNCS, vol. 1788, pp. 20–28. Springer, Heidelberg (2000)
10. Kikuchi, H.: (m+1)st-price auction protocol. In: Proceedings of the 5th International Conference on Financial Cryptography, London, UK, pp. 351–363. Springer, Heidelberg (2002)
11. Hausheer, D., Stiller, B.: Peermart: The technology for a distributed auction-based market for peer-to-peer services. In: Proceedings of the IEEE International Conference on Communications, vol. 3, pp. 1583–1587 (2005)
12. Haque, N., Jennings, N.R., Moreau, L.: Scalability and robustness of a network resource allocation system using market-based agents. Netnomics 7(2), 69–96 (2005)
13. Ramasubramanian, V., Sirer, E.G.: Perils of transitive trust in the domain name system. Technical Report TR2005-1994, Cornell University, Ithaca, New York, USA (2005)
14. Kuwabara, K., Ishida, T., Nishibe, Y., Suda, T.: An equilibratory market-based approach for distributed resource allocation and its applications to communication network control. In: Clearwater, S.H. (ed.) Market-Based Control: A Paradigm for Distributed Resource Allocation, pp. 53–73. World Scientific, Singapore (1996)
15. Midgley, D.F., Marks, R.E., Cooper, L.G.: Breeding competitive strategies. Management Science 43(3), 257–275 (1997)
16. Cheung, Y., Bedingfield, S., Huxford, S.: Monitoring and interpreting evolved behaviours in an oligopoly. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 697–701 (1997)
17. Mas-Colell, A., Whinston, M.D., Green, J.R.: Micro-Economic Theory. Oxford University Press, Oxford (1995)
18. Kephart, J.O., Hanson, J.E., Sairamesh, J.: Price-war dynamics in a free-market economy of software agents. In: Proceedings of the Sixth International Conference on Artificial Life, pp. 53–62. MIT Press, Cambridge (1998)

GA-Net: A Genetic Algorithm for Community Detection in Social Networks

Clara Pizzuti

ICAR-CNR,
Via P. Bucci 41C, 87036 Rende (CS), Italy
pizzuti@icar.cnr.it

Abstract. The problem of community structure detection in complex networks has been intensively investigated in recent years. In this paper we propose a genetic based approach to discover communities in social networks. The algorithm optimizes a simple but efficacious fitness function able to identify densely connected groups of nodes with sparse connections between groups. The method is efficient because the variation operators are modified to take into consideration only the actual correlations among the nodes, thus sensibly reducing the research space of possible solutions. Experiments on synthetic and real life networks show the capability of the method to successfully detect the network structure.

1 Introduction

The suitability of networks to represent many real world systems has given an impressive spur to the recent research area of complex networks. Collaboration networks, the Internet, the world-wide-web, biological networks, communication and transport networks, social networks are just some examples. Networks, in general, are constituted by a set of objects and by a set of interconnections among these objects. In social networks the objects are people and the connections represent social relations, such as common interests, friendship, religion, and so on. An interesting property to investigate, typical to many networks, is the *community structure*, i.e. the division of networks into groups (also called clusters) having dense intra-connections, and sparse inter-connections. The capability of detecting the partitioning of a network in clusters can give important information and useful insights to understand how the structure of ties affects individuals and their relationships. The problem of community detection has been receiving a lot of attention and many different approaches have been proposed [10,16,18,4,20,2,11].

In this paper we propose a new algorithm, named *GA-Net*, to discover communities in networks by employing genetic algorithms. The approach introduces the concept of *community score* to measure the quality of a partitioning in communities of a network, and tries to optimize this quantity by running the genetic algorithm. All the dense communities present in the network structure are obtained at the end of the algorithm by selectively exploring the search space, without the need to know in advance the exact number of groups. Specialized variation operators allow to reduce the space of the possible solutions thus improving the convergence of the method. The main novelties of the approach can be summarized as follows. The concept of *community score*, that provides a global quality measure of a partitioning in communities, is defined. The notion of $safe$ individual is introduced to avoid useless computation to the genetic algorithm and specialized variation operators that generate safe individuals are employed. Unlike many existing methods, the algorithm does not require the number of communities to find. This number is automatically determined by the optimal value of the *community score*. Experiments on synthetic and real life networks show the capability of the genetic approach to correctly detect communities with results comparable with state-of-the-art approaches.

The paper is organized as follows. The next section provides the necessary background to formalize the problem and defines the quality metric. In section 3 a description of the representation adopted and the variation operators used is provided. In section 4 an overview of the main proposals in community detection algorithms is given. In section 5, finally, the results of the method on synthetic and real life data sets are presented.

2 Problem Definition

A social network $\mathcal{SN}$ can be modelled as a graph $G = (V, E)$ where V is a set of objects, called nodes or vertices, and E is a set of links, called edges, that connect two elements of V. A community (or cluster) in a network is a group of vertices having a high density of edges within them, and a lower density of edges between groups. The problem of detecting k communities in a network, where the number k is unknown, can be formulated as finding a partitioning of the the nodes in k subsets that are highly intra-connected and sparsely inter-connected. To deal with graphs, often the adjacency matrix is used. If the network is constituted by N nodes, the graph can be represented with the $N \times N$ adjacency matrix A, where the entry at position (i, j) is 1 if there is an edge from node i to node j, 0 otherwise. The problem of detecting communities in a network can then be transformed to that of finding a partitioning of A in k sub-matrices that maximize the sum of densities of the sub-matrices. A naive density measure for a sub-matrix of n rows/columns is the number of ones (i.e. interactions) it contains. The higher the number of ones, the more connected the n nodes. However, counting the number of interactions does not give any information about the interconnections among the nodes. A density measure based on volume and row/column means, allowing to detect maximal and dense sub-matrices, has been introduced in [1], and applied to find Co-clusters in sparse binary matrices. Co-clustering[13], also known as bi-clustering, differently from clustering, tries to simultaneously group both the dimensions (objects and features) of a data set. Sub-matrix identification can be considered as a special case of co-clustering in which the two dimensions represent the same concept, i.e. the nodes of the graph. In the following the density measure used in [1], specialized for adjacency matrices, is recalled, and the new concept of *community score*, that gives a global measure of the network partitioning in clusters, is defined. In the following, without loss of generality, we assume an undirected graph. This assumption implies that the adjacency matrix is symmetric.

Let $S = (I, J)$ be sub-matrix of A, where I is a subset of the rows $X = \{I_1, \ldots, I_N\}$ of A, and J is a subset of the columns $Y = \{J_1, \ldots, J_N\}$ of A.

Let a_{iJ} denote the *mean value* of the ith row of the S, and a_{Ij} the mean of the jth column of S. More formally,

$$a_{iJ} = \frac{1}{|J|}\sum_{j\in J} a_{ij}, \text{ and } a_{Ij} = \frac{1}{|I|}\sum_{i\in I} a_{ij}$$

The *volume* v_S of a sub-matrix $S = (I, J)$ is the number of 1 entries a_{ij} such that $i \in I$ and $j \in J$, that is $v_S = \sum_{i\in I, j\in J} a_{ij}$.

Given a sub-matrix $S = (I, J)$, the *power mean of* S *of order* r, denoted as $\mathbf{M}(S)$ is defined as

$$\mathbf{M}(S) = \frac{\sum_{i\in I}(a_{iJ})^r}{|I|}$$

A measure based on volume and row/column mean, that allows the detection of maximal and dense sub-matrices, can be defined as follows. Given a sub-matrix $S = (I, J)$, let $\mathbf{M}(S)$ be the power mean of S of order r. The *score* of S is defined as $Q(S) = \mathbf{M}(S) \times v_S$. The *community score* of a partitioning $\{S_1, \ldots S_k\}$ of A is defined as

$$\mathcal{CS} = \sum_{i}^{k} Q(S_i)$$

The problem of community identification can be formulated as the problem of maximize $\mathcal{CS}$. It is worth to note that higher values of the exponent r bias the CS towards matrices containing a low number of zeroes. In fact, it amplifies the weight of the densely interconnected nodes, while reducing those of less connected in the computation of the *community score*. In the experimental result section we show that when the modular structure of the network is not well defined, higher values of r help in detecting communities.

3 Genetic Representation and Operators

In this section we give a description of the algorithm *GA-Net*, the representation adopted for partitioning the network, and the variation operators used.

Genetic representation. Our clustering algorithm uses the locus-based adjacency representation proposed in [19] and employed by [9,14] for multiobjective clustering. In this graph-based representation an individual of the population consists of N genes $g_1, \ldots, g_N$ and each gene can assume allele values j in the range $\{1, \ldots, N\}$. Genes and alleles represent nodes of the graph $G = (V, E)$ modelling a social network $\mathcal{SN}$, and a value j assigned to the ith gene is interpreted as a link between the nodes i and j of V. This means that in the clustering solution found i and j will be in the same cluster. A decoding step, however, is necessary to identify all the components of the corresponding graph. The nodes participating to the same component are assigned to one cluster. As observed in [9], the decoding step can be done in linear time. A main advantage of this representation is that the number k of clusters is automatically determined by the number of components contained in an individual and determined by the decoding step. Suppose to have the network shown in figure 1(a). It consists of eleven nodes numbered from 1 to 11. The network can be partitioned in the three groups visualized by different colors and shapes of the nodes. Out of the many possible genotypes, that showed in figure 1(b), corresponding to the optimal solution, is translated in the graph structure given in figure 1(c). Each connected component provides a grouping of nodes that corresponds to the partitioning of the network in figure 1(a).

Objective Function. As described above, the decoding of an individual provides a different number k of components $\{S_1, \ldots S_k\}$ in which the graph is partitioned. We are interested in identifying a partitioning that optimizes the *community score* because, as already discussed in the previous section, this guarantees highly intra-connected and sparsely inter-connected communities. The objective function is thus $\mathcal{CS} = \sum_i^k Q(S_i)$

Initialization. Our initialization process takes in account the effective connections of the nodes in the social network. A random generation of individuals could generate components that in the original graph are disconnected. In fact, a randomly generated individual could contain an allele value j in the ith position, but no connection exists between the two nodes i and j, i.e. the edge (i, j) is not present. In such a case it is obvious that grouping in the same cluster both nodes i and j is a wrong choice. In order to overcome this drawback, once an individual is generated, it is $repaired$, that is a check is executed to verify that an effective link exists between a gene at position i and the allele value j. This value is maintained only if the edge (i, j) exists. Otherwise, j is substituted with one of the neighbors of i. This guided initialization biases the algorithm towards a decomposition of the network in connected groups of nodes. We call an individual generating this kind of partitioning $safe$ because it avoids uninteresting divisions containing unconnected nodes. $Safe$ individuals improve the convergence of the method because the space of the possible solutions is restricted.

Uniform Crossover. We used uniform crossover because it guarantees the maintenance of the effective connections of the nodes in the social network in the child individual. In fact, because of the biased initialization, each individual in the population is safe, that is it has the property, that if a gene i contains a value j, then the edge (i, j) exists. Thus, given two safe parents, a random binary vector is created. Uniform crossover then selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form the child. The child at each position i contains a value j coming from one of the two parents. Thus the edge (i, j) exists. This implies that from two safe parents a safe child is generated. Figure 1 on the right shows an example of crossover. Two parents, individuals A and B, and their graph-based representations are reported. Uniform crossover of A and B gives the child C.

Mutation. The mutation operator that randomly change the value j of a i-th gene causes a useless exploration of the search space, because of the same above observations on node connections. Thus the possible values an allele can assume are restricted to the neighbors of gene i. This $repaired$ mutation guarantees the generation of a safe mutated child in which each node is linked only with one of its neighbors.

Given a network $\mathcal{SN}$ and the graph G modelling it, *GA-Net* starts with a population initialized at random and $repaired$ to produce $safe$ individuals. Every individual generates a graph structure in which each component is a connected subgraph of G. For a fixed number of generations the genetic algorithm computes the fitness function of each solution member, and applies the specialized variation operators to produce the new population. In the experimental result section we show that the fitness function guides the genetic algorithm to successfully identify the best partitioning of $\mathcal{SN}$,

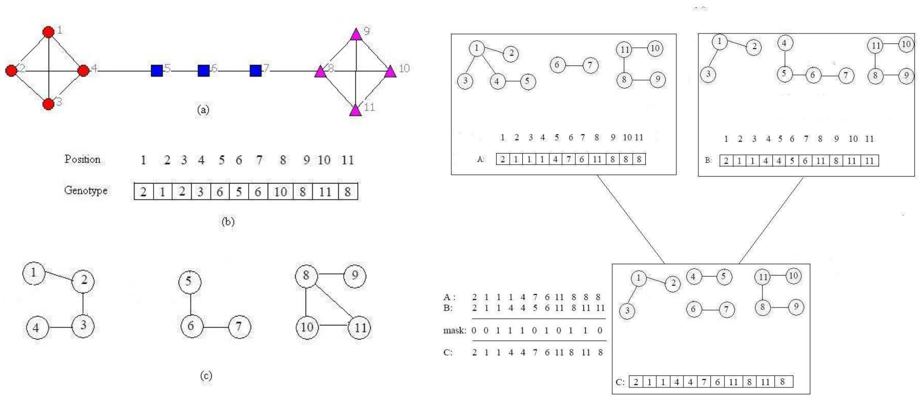

(a)
Position 1 2 3 4 5 6 7 8 9 10 11
Genotype 2 1 2 3 6 5 6 10 8 11 8
(b)
(c)
A: 2 1 1 1 4 7 6 11 8 8 8
B: 2 1 1 4 4 5 6 11 8 11 11
mask: 0 0 1 1 1 0 1 0 1 1 0
C: 2 1 1 4 4 7 6 11 8 11 8

Fig. 1. (a) A network modelled as a graph; (b) the locus-based representation of a genotype; (c) the graph-based structure of the genotype. On the right Uniform crossover of two individuals, their genotype, their graph-based representation, and the child generated.

converging in a few iterations to the solution. Before presenting the experiments, in the next section an overview of the main approaches to community detection is given.

4 Related Work

Many different algorithms, coming from different fields such as physics, statistics, data mining, have been proposed to detect communities in complex networks [8,10,16,18,4,20,17,2,11]. In the following we review some of the most known.

One of the most famous algorithm has been presented by Newman and Girvan [8,18]. The method is a divisive hierarchical clustering method [6] based on an iterative removal of edges from the network. The edge removal splits the network in communities. The edges to remove are chosen by using *betweenness* measures. The idea underlying the edge betweenness comes from the observation that if two communities are joined by a few inter-community edges, then all the paths from vertices in one community to vertices in the another must pass through these edges. Paths determine the betweenness score to compute for the edges. By counting all the paths passing through each edge, and removing the edge scoring the maximum value, the connections inside the network are broken. This process is repeated, thus dividing the network into smaller components until a stop criterion is reached. The criterion adopted to stop the division is the *modularity*. Given k communities, the modularity is defined as follows. Let e_{ij} be the fraction of edges in the network connecting vertices from group i to those of group j, and $a_i = \sum_j e_{ij}$. Then $M = \sum_i (e_{ii} - a_i^2)$ is the fraction of edges inside communities minus the expected value of the fraction of edges if edges fall at random without regard to the community structure. Values approaching 1 indicate strong community structure. Thus the algorithm computes the modularity of all the clusters obtained by applying the hierarchical approach, and returns as result the clustering having the highest value of modularity. An agglomerative variant of this approach is presented in [16], and a faster method version, based on the same strategy, is described in [4].

Hopcroft et al. [10] present an agglomerative hierarchical method for clustering large linked networks to identify stable or natural cluster. A cluster is deemed natural if it appears in the clustering process when a given percentage of links are removed.

Radicchi et al. [20] propose two quantitative definitions of community and an algorithm to identify communities. The quantitative definitions of community are based on the degree of a node. A subgraph is a *community in strong sense* if each node has more connections within the community than the rest of the graph. A subgraph is a *community in a weak sense* if the sum of all in-degrees in V is greater than the sum of the out-degrees. The algorithm is a divisive hierarchical method based on the concept of *edge-clustering coefficient*, defined in analogy with the node clustering coefficient [15], as the number of triangles an edge participates, divided by the number of triangles it might belong to, given the degree of the adjacent nodes. Their algorithm works like that of Newmann and Girvan, the difference being that instead of choosing to remove the edge with the highest edge betweenness, the removed edges are those having the smallest value of edge-clustering coefficient.

Approaches to community detection based on Genetic Algorithms can be found in [21,22,7]. In [21,22] the authors present a genetic algorithm that uses as fitness function the network modularity proposed by Newmann and Girvan. An individual is constituted by N genes, where N is the number of objects. The ith gene corresponds to the ith node, and its value is the community identifier of node i. They use a non standard one-way crossover operation in which, given two individuals A and B, a community identifier j is chosen at random, and the identifier j of the nodes $j_1, \ldots j_h$ of A is transferred to the same nodes of B.

A different approach is described in [7] where a random walk distance measure between graphs is integrated in a genetic algorithm to cluster social networks. The representation they use is the k-medoids where each cluster center is represented by one of the nodes of the network. Of course this means that the number k of clusters must be known in advance. The fitness function tries to minimize the sum of all the pair-wise distances between nodes.

5 Experimental Results

In this section we study the effectiveness of our approach on a synthetic data set. Then we compare the results obtained by *GA-Net* with those reported by Girvan and Newman in [8,18,17] on some real-worlds networks for which the partitioning in communities is known. In both cases we show that our genetic algorithm successfully detects the network structure and is competitive with that of Girvan and Newman. The *GA-Net* algorithm has been written in MATLAB, using the Genetic Algorithms and Direct Search Toolbox 2. The experiments have been performed on a Pentium 4 machine, 1800MHz, 1GB RAM. We employed standard parameters for the genetic algorithm, crossover rate 0.8, mutation rate 0.2, elite reproduction 10% of the population size, roulette selection function. The population size was 300, the number of generations 30.

Synthetic data set. In order to check the ability of our approach to successfully detect the community structure of a network, we use the benchmark proposed by Girvan and Newan in [8]. The network consists of 128 nodes divided into four communities of 32 nodes each. Edges are placed between vertex pairs at random but such that $z_{in} + z_{out} = 16$, where z_{in} and z_{out} are the internal and external degree of a node with respect to its community. If $z_{in} > z_{out}$ the neighbors of a node inside its group are more than the neighbors belonging to the other three groups, thus a good algorithm should discover them. We generated 50 different networks for values of z_{out} ranging from 0 to 8, and used the *Normalized Mutual Information* to measure the similarity between the true partitions and the detected ones. The *Normalized Mutual Information* is a similarity measure proved to be reliable by Danon et al. [5]. Given two partitions A and B of a network in communities, let C be the confusion matrix whose element C_{ij} is the number of nodes of community i of the partition A that are also in the community j of the partition B. The normalized mutual information $I(A, B)$ is defined as :

$$I(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} C_{ij} log(C_{ij}N/C_{i.}C_{.j})}{\sum_{i=1}^{c_A} C_{i.} log(C_{i.}/N) + \sum_{j=1}^{c_B} C_{.j} log(C_{.j}/N)}$$

where c_A (c_B) is the number of groups in the partition A (B), $C_{i.}$ ($C_{.j}$) is the sum of the elements of C in row i (column j), and N is the number of nodes. If $A = B$, $I(A, B) = 1$. If A and B are completely different, $I(A, B) = 0$.

Figure 2(a) shows the normalized mutual information, averaged over the 50 runs, for different values of the exponent r when the external degree z_{out} increases from 0 to 8. The figure point out that until $z_{out} \leq 5$ the algorithm is successful in detecting the true communities in almost more than 80% of cases, independently the value of r. However, as soon as the network fuzziness increases, in order to discover at least 50% of the true groups the parameter r plays an important role. In fact, the higher the number of interconnections, the more indistinguishable the network structure because communities are mixed with each other, but augmenting the value of r the algorithm is still able to identify the hidden groups.

Real-life data set. We now show the application of *GA-Net* on three real-world networks, the *American College Football*, *Krebs' books on American politics*, *Bottlenose*

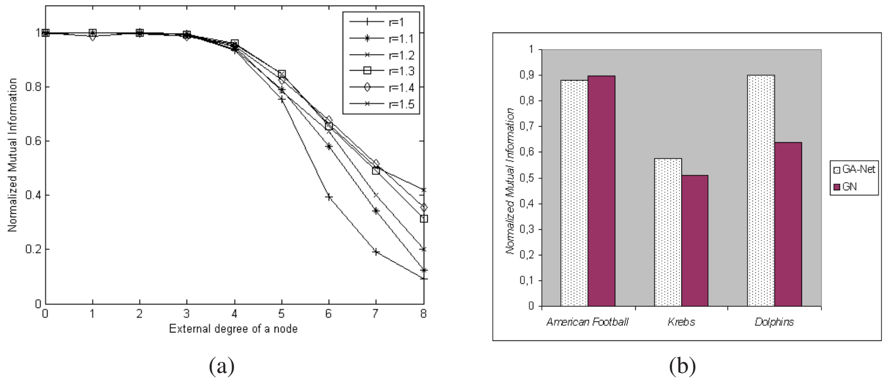


Fig. 2. (a): Normalized mutual information obtained by *GA-Net* on the synthetic data set for different values of the exponent r. (b): Comparison of GA-Net and Girvan and Newman's (denoted GN) algorithms relative to Normalized Mutual Information for American College Football, Krebs'political books, Dolphins data sets.

Dolphins, well studied in the literature and compare our results with those obtained by Girvan and Newman in [8,18,17]. The American College Football network [8] comes from the United States college football. The network represents the schedule of Division I games during the 2000 season. Nodes in the graph represent teams and edges represent the regular season games between the two teams they connect. The teams are divided in conferences. The teams on average played 4 inter-conference matches and 7 intra-conference matches, thus teams tend to play between members of the same conference. The network consists of 115 nodes and 616 edges grouped in 12 teams. The network of political books was compiled by V. Krebs. The nodes represent 105 recent books on American politics brought from Amazon.com, and edges join pairs of books frequently purchased by the same buyer (unpublished http://www.orgnet.com/). Books were divided by Newman [17] according to their political alignment (conservative or liberal), except for a small number of books (13) having no clear affiliation. The last example is the social network of 62 bottlenose dolphins living in Doubtful Sound, New Zealand, compiled by Lusseau [12] from seven years of dolphins behavior. A tie between two dolphins was established by a their statistically significant frequent association. The network split naturally into two large groups, the number of ties being 159.

For each network we computed the normalized mutual information on the base of the results reported by the authors. Regarding *GA-Net*, we run it 10 times and computed the average normalized mutual information over these 10 runs. The results are reported in figure 2(b). The figure clearly shows the very good performance of *GA-Net* with respect to Girvan and Newman's approach. In fact, on the American College Football network, *GA-Net* obtained an average normalized mutual information of 0.8825 over the 10 runs, with a worst value of 0.8417, and a best value of 0.9031, while the result of [8] was 0.8957. For the Krebs' network Newman [17] obtained 0.5107, while our approach 0.5756. Finally, on the dolphin network Girvan and Newman [18] obtained a normalized mutual information of 0.64. *GA-Net* over 10 runs obtained an average value of 0.8992. It is worth to point out that for 7 out of the 10 runs, *GA-Net* misplaced just node 40, which is connected to only two nodes: node 37 which belongs to the first group, and node 58 that belongs to the second group, thus the membership to one of the two communities is indistinguishable without adding semantics to the kind of link interconnecting the dolphins. In one over the 10 runs *GA-Net* obtained the exact partitioning of dolphins. To conclude figure 3 reports the result of running our algorithm on *Zackary's Karate Club Study.* This network was generated by Zachary [23], who studied the friendship of 34 members of a karate club over a period of two years. During this period, because of disagreements, the club divided in two groups almost of the same size. The figure has been reproduced by using the $NetDraw$ software [3]. We found four groups, depicted in the figure by different colors and shapes of the nodes. However, the two small communities are each a subgroup of the two effective communities. Thus our algorithm is able to detect more compact interactions. For example, as pointed out on the figure, the small community on the bottom left is characterized by five nodes each of which is connected to the bigger community only through the friendship to node 1, while among these five nodes a tighter connection exists. Girvan and Newman in [8] found the two groups in which the karate club divided. Node 3, however, was misplaced. A similar result is reported in [21]. In this latter approach the node 10 is

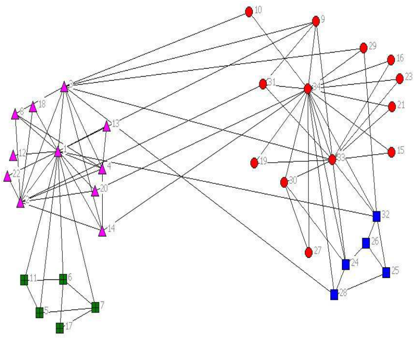


Fig. 3. Community structure found by the genetic based method *GA-Net*

misplaced. Our approach, on the contrary, is able to correctly classify both these nodes. The results obtained show the capability of genetic algorithms to effectively deal with community identification in networks.

6 Conclusions

The paper presented a genetic algorithm for detecting communities in social networks. The approach introduces the concept of community score, and searches for an optimal partitioning of the network by maximizing the community score. All the dense communities present in the network structure are obtained at the end of the algorithm by selectively exploring the search space, without the need to know in advance the exact number of groups. The concept of community score, though simple, revealed very efficacious. In fact, experiments on synthetic and real life networks showed the capability of the genetic approach to correctly detect communities with comparable results with state-of-the-art approaches. Future research will aim at applying multi-objective optimization to improve quality results.

References

1. Angiulli, F., Cesario, E., Pizzuti, C.: A greedy search approach to co-clustering sparse binary matrices. In: Proceedings of the 18th International Conference on Tools with Artificial Intelligence (ICTAI 2006), pp. 363–370 (2006)
2. Arenas, A., Diaz-Guilera, A.: Synchronization and modularity in complex networks. European Physical Journal ST 143, 19–25 (2007)
3. Borgatti, S.P.: Netdraw 1.0: Network visualization software. Harvard: Analytic technologies (2002)
4. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E 70, 066111 (2004)
5. Danon, L., Díaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. Journal of Statistical Mechanics, P09008 (2005)
6. Dubes, R.C., Jain, A.K.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)

7. Firat, A., Chatterjee, S., Yilmaz, M.: Genetic clustering of social networks using random walk. Computational Statistics and Data Analysis 51, 6285–6294 (2007)
8. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. National. Academy of Science. USA 99, 7821–7826 (2002)
9. Handle, J., Knowles, J.: An evolutionary approach to multiobjective clustering. IEEE transactions on Evolutionary Computation 11(1), 56–76 (2007)
10. Hopcroft, J.E., Khan, O., Kulis, B., Selman, B.: Natural communities in large linked networks. In: Proc. International Conference on Knowledge Discovery and Data Mining (KDD 2003), pp. 541–546 (2003)
11. Lozano, S., Duch, J., Arenas, A.: Analysis of large social datasets by community detection. European Physical Journal ST 143, 257–259 (2007)
12. Lusseau, D.: The emergent properties of dolphin social network. Biology Letters, Proc. R. Soc. London B (suppl.) (2003)
13. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: A survey. IEEE Transactions on Computational Biology and Bioinformatics 1(1), 24–45 (2004)
14. Makate, N., Miki, M., Hiroyasu, T., Senda, T.: Multiobjective clustering with automatic k-determination fot large-scale dara. In: Proc. of the Int. Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 861–868 (2007)
15. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
16. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Physical Review E 69, 066133 (2004)
17. Newman, M.E.J.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA 103, 8577–8582 (2006)
18. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69, 026113 (2004)
19. Park, Y.J., Song, M.S.: A genetic algorithm for clustering problems. In: Proc. of 3rd Annual Conference on Genetic Algorithms, pp. 2–9 (1989)
20. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proc. Natl. Acad.Sci. USA (PNAS 2004) 101(9), 2658–2663 (2004)
21. Tasgin, M., Bingol, A.: Communities detection in complex networks using genetic algorithms. In: Proc. of the European Conference on Complex Systems (ECSS 2006) (2006)
22. Tasgin, M., Herdagdelen, A., Bingol, A.: Communities detection in complex networks using genetic algorithms. oai:arXiv.org:0711.0491v1 [physics.soc-ph] (2007)
23. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33, 452–473 (1977)

Learning Walking Patterns for Kinematically Complex Robots Using Evolution Strategies

Malte Römmermann[1], Mark Edgington[2], Jan Hendrik Metzen[1], Jose de Gea[2], Yohannes Kassahun[2], and Frank Kirchner[1,2]

[1] German Research Center for Artificial Intelligence (DFKI)
Robert-Hooke-Str. 5, D-28359, Bremen, Germany
[2] Robotics Group, University of Bremen
Robert-Hooke-Str. 5, D-28359, Bremen, Germany

Abstract. Manually developing walking patterns for kinematically complex robots can be a challenging and time-consuming task. In order to automate this design process, a learning system that generates, tests, and optimizes different walking patterns is needed, as well as the ability to accurately simulate a robot and its environment. In this work, we describe a learning system that uses the CMA-ES method from evolutionary computation to learn walking patterns for a complex legged robot. The robot's limbs are controlled using parametrized distorted sine waves, and the evolutionary algorithm optimizes the parameters of these waveforms, testing the walking patterns in a physical simulation. The best solutions evolved by this system has been transferred to and tested on a real robot, and has resulted in a gait that is superior to those previously designed by a human designer.

1 Introduction

Locomotion control of legged systems is a far more challenging task than locomotion of wheeled systems. Classical approaches to robot control that use (online) trajectory planning are not very well suited for the control of walking robots [15] since these approaches require a highly accurate model of the robot and its environment in order to compute the precise trajectories of a robot's leg movements. In contrast to these classical methods, biologically inspired robot control methods are easier to implement and require less computational time [2]. This is due to the fact that they do not require sophisticated models of the environment, but generate the walking pattern based on a set of predetermined elementary patterns.

Unfortunately, manually designing these patterns for a kinematically complex robot is a difficult and time-consuming task that can only be done in a trial-and-error fashion. After an appropriate gait has been designed, even small changes to the morphology of the robot can cause this gait to become unstable, requiring repetition of the design process. Furthermore, manual design is susceptible to the *designer bias*: human designers might tend to focus on certain promising areas in the space of solutions, while ignoring possibly more efficient but counterintuitive solutions in other areas.

Because of these difficulties associated with manual design, the use of autonomous or at least semi-autonomous approaches for designing walking patterns is very attractive. Such approaches relieve the designer from much of the drudgery involved in the design process, and reduce designer bias, potentially allowing the designer to find solutions that exceed the performance of manual solutions. Evolutionary algorithms present a promising approach in this domain. They can search complex spaces without being easily trapped in local extrema, and at the same time they are able to cover broad parts of the search space [13]. Evolutionary algorithms are especially suited for the optimization of a set of real-valued parameters. One way of combining evolutionary algorithms with robot locomotion is to describe the movements of the individual joints of the robot by parameterized periodical curves. The combined rhythmic oscillations of the single joints described by these curves results in entire leg movements, and thus, potentially in a walking pattern. In this work, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [6] is used as the evolutionary algorithm, and the periodic curves are described with distorted sine waves.

This paper is organized as follows. First, a brief review of works done in the area of evolving/learning walking patterns is presented. Next, the experimental platform used in this work is described. Following this, the methodology used for learning walking patterns is presented, which involves both the evolutionary algorithm and the parametrization of the joint curves. We then introduce and discuss two experiments that were performed using this learning methodology, and finally present conclusions and further areas of work.

2 Review of Works

Among the first works on the evolution of walking patterns is the work of Beer and Gallagher [1]. In their work, artificial evolution was used to evolve a static gait for a simulated, kinematically simple six-legged robot. Each leg was controlled by a five neuron, fully connected, continuous-time neural network, and the corresponding neurons of the controllers of the neighboring legs were connected. The weights of the six controllers were constrained to be identical, yielding a total of 50 parameters to be optimized. The genetic algorithm they used was able to evolve a tripod gait in all conducted runs. Lewis et al. [10] transferred the approach of Beer and Gallagher for the first time onto a physical robot. While Lewis et al. did not perform the evolution itself in the real world (but only tested the evolved gait on a physical robot), Gomi and Ide [3] evolved a gait for an octopod robot completely online. While in most experiments the control architecture was fixed, Gruau and Quatramaran [5] used the cellular encoding to evolve both the weights and the structure of a controller for an octopod robot.

In contrast to the approaches discussed above, in which a static gait was evolved, Hornby [7] and Röfer [12] evolved a dynamic gait for AIBO[1] robots.

[1] AIBO: Very popular quadruped robot from Sony.

They used a genetic algorithm to optimize the parameters, and evolution was performed on-line on real robots like in the work of Gomi and Ide. Zhang [17] achieved recently the fastest gait for these robots by combining a learning method with evolutionary techinques. Hornby used the central pattern generator (CPG) [4] model for the controller, which allow oscillatory patterns to be generated in the absence of external stimuli. This model was also used by Reil and Husbands [11], who used evolutionary methods to optimize the parameters of CPGs on a simulated, bipedal robot, and show that the quality of an evolutionary method depends heavily on the chosen fitness function. However, few works deal with more than two or three degrees of freedom per leg. Ito [8] used evolutionary techniques in a redundant, multi-legged robot, although only simulated results were presented.

3 Robotic Platform

3.1 Hardware

The robot used in our work, named SPOT, was developed at the University of Bremen's robotics laboratory. It is a kinematically complex robot whose four legs are made up of a total of 24 independently controllable joints. The robot, which weighs about 30kg, shares many similarities with the ARAMIES robot [16], both in its physical construction, and in its software architecture. Its six degrees of freedom per leg make the manual generation of walking patterns extremely complex and time-consuming. On the other hand, due to the legs' complexity, they can be used not only for traversing a variety of terrains, but also for manipulating objects [9]. In contrast to the ARAMIES robot, SPOT also possesses a hip joint which enables it to bend at the waist. This makes it possible, for example, for the robot to sit on its hind legs, or to climb steep slopes.

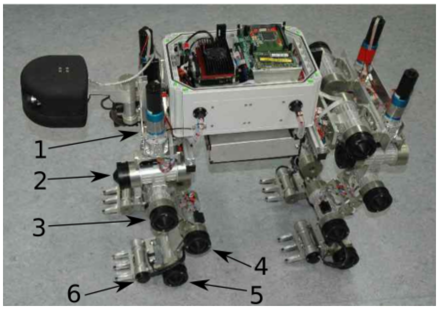
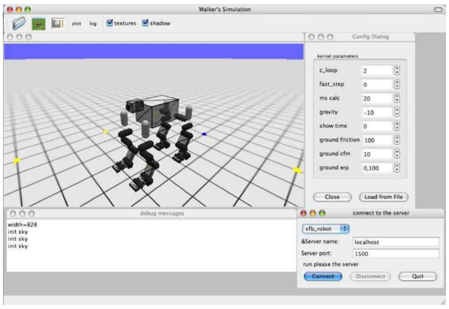


Fig. 1. SPOT: The robot used for our experiments. The joints of the front left leg are labeled with numbers in the picture.



textures ✓ shadow ✓

kernel parameters

c_loop	2
fast_step	0
ms calc	20
gravity	-10
show time	0
ground friction	100
ground cfm	10
ground erp	0,100

```
width=828
init sky
init sky
init sky
```

sfb_robot
&Server name: localhost
Server port: 1500
run please the server

Fig. 2. The "WalkerSim" simulation environment

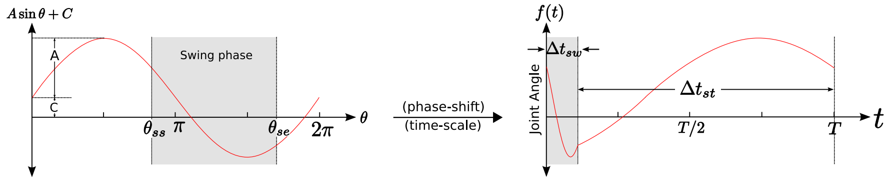


Fig. 3. Construction of a joint-angle pattern waveform. Six parameters are used to represent the waveform: offset (C), amplitude (A), angular swing-start (θ_{ss}), angular swing-end (θ_{se}), swing-duration (Δt_{sw}), and stance-duration (Δt_{st}). The region of the sine wave (left graph) designated as the swing-phase is shifted left to $\theta = 0$. This shifted function is further time-scaled so that the swing and stance phases are compressed or expanded to fit into the Δt_{sw} and Δt_{st} time durations (right graph).

3.2 Simulation Environment

The evolutionary process makes use of the "WalkerSim" simulation environment, developed at the University of Bremen's robotics laboratory. The physical engine used by the software is the Open Dynamics Engine (ODE) [14]. The simulation software provides an interface between walking robots and ODE, and a set of sensors that can be used by a simulated robot. The joint (motor) models have been carefully designed to approximate the behavior of the real motors used by SPOT, while not being too computationally complex to simulate. Figure 2 shows a screenshot of "WalkerSim" in which a model of SPOT is shown.

4 Learning Methodology

4.1 CPG Representation

In this work, the function describing the joint-angle patterns of SPOT's leg joints is represented in the form of a distorted sine wave. Figure 3 shows how this joint-angle function is constructed. A basic sine function with offset C and amplitude A over the domain $0 \leq \theta < 2\pi$ is shown, whose range is considered to represent the angular-position of a robot joint. An angular-interval $[\theta_{ss}, \theta_{se}]$ is chosen within this sine function's domain, where θ_{ss} and θ_{se} represent the beginning and end, respectively, of the swing-phase of a robot leg. This basic sine function is phase shifted such that the swing-phase begins at $\theta = 0$. Finally, this phase-shifted function is time-scaled, and the interval $[0, \Delta\theta_{sw}]$ (where $\Delta\theta_{sw} = \theta_{se} - \theta_{ss}$) of the phase-shifted function is compressed or expanded to a new corresponding interval $[0, \Delta t_{sw}]$ of the distorted waveform, where Δt_{sw} is the desired duration (in seconds) of the swing-phase of the robot leg. In the same way, the interval $[\Delta\theta_{sw}, 2\pi]$ is compressed or expanded to a corresponding interval $[\Delta t_{sw}, \Delta t_{sw} + \Delta t_{st}]$ of the distorted waveform, where Δt_{st} is the desired duration of the stance-phase of the robot leg.

4.2 CMA-ES

Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [6] is an advanced form of evolution strategy [13] that can perform efficient optimization even with small population sizes. Each individual is represented by an $n-$dimensional real valued solution vector.

CMA-ES does not provide a means of bounding the search space (i. e. the object variables are optimized on $\mathbb{R}$). Therefore, to deal with the parameter-value limits imposed by the robot hardware, we chose to use a function that projects the whole space of real values onto a range that the robot-controller can handle (i. e. the *control-value space*). The projection function $P : \mathbb{R} \longrightarrow [x_{offset}, x_{offset} + x_{max}]$ is defined by the chain $x \longrightarrow x' \longrightarrow x'' \longrightarrow x^*$ of the following transformations:

$$x' = x \bmod 2x_{max} \tag{1}$$

$$x'' = \begin{cases} x', & \text{if } x' \leq x_{max} \\ 2x_{max} - x', & \text{if } x' > x_{max} \end{cases} \tag{2}$$

$$x^* = x'' + x_{offset} \tag{3}$$

This projection function is continuous, which is important for the learning algorithm. It is also chosen such that the absolute value of its derivative is constant wherever this derivative exists. This is done to avoid the negative effects on the performance of the learning algorithm that can result from nonlinearly projecting the learned values onto the control-value space. Such a nonlinear projection like a modulo function could cause the fitness landscape to become more fragmented, making it more difficult for an evolutionary method to find a global maximum.

4.3 Evaluation of Individuals

An individual is represented as a vector of parameters, that have to be optimized. An individual's values are determined by the evolutionary algorithm (CMA-ES). These real-valued variables are mapped to a suitable range for each parameter that is used to describe the joint patterns. To evaluate an individual, the patterns that the individual specifies are tested in the simulation for a period of 10 seconds. The evaluation time is choosen empherically and found to be a good balance between computational cost and adequate evaluation of individuals. At the beginning of each evaluation, the simulated robot starts at the same initial position. The simulation proceeds, and the rewards (e.g. torque, load, and distance traveled) are calculated at the end of the simulation testing period. From these rewards a fitness value is calculated and returned to the evolutionary algorithm.

5 Experiments

In this section two experiments are presented and their results are discussed. The first experiment investigates the feasibility of the approach, and the possibility of transferring the simulated behaviors to the real robot. In this experiment, the complexity of the learning problem is reduced by using empirical knowledge.

In the second experiment, the learning algorithm optimizes a larger number of parameters, which results in a larger search space for the evolutionary method and greater challenges in transfering the results to the real robot.

In both experiments, each left leg is made to execute the same patterns that the corresponding right leg executes (including a phase shift between the legs). This restriction is made because of the symmetry of the robot architecture, and has the advantage that patterns for only one side of the robot need to be learned. As explained in Section 4.1, each pattern is defined by the four parameters A, C, θ_{se}, and θ_{ss}, which must be optimized. The other parameters that affect the walking behavior are the CPG waveform period T, the duration of the swing phase Δt_{sw}, and the phase shifts between the walking patterns of each leg.

In order to assess the evolved walking patterns, a fitness value is used which is based on the following variables which are measured in the simulation: The distance traveled d, the average torque τ and the average load l of the joints, a boolean value τ' that indicates that the maximal torque of the toe joints is above a certain threshold, and a boolean value g that indicates if parts of the robot other than the feets has ground contact. The fitness function for both experiments is defined as sum of four terms. The first term $\frac{-\tau}{d}$ assesses the energy efficiency, i. e. the consumed energy for the traveled distance. The second term $\frac{-l}{d}$ is based on the load incurred on the joints over the travelled distance. The third term $-1000g$ checks whether parts of the robot other than the feets come into contact with the ground and a strong negative reward is added if that is the case. The last term $1000\tau'$ checks whether the maximal torque of one toe joint exceeds a certain threshold and a strong negative reward is added if that is the case. The resulting fitness function is described by the following equation:

$$fitness = \frac{-\tau}{d} + \frac{-l}{d} - 1000g - 1000\tau' \tag{4}$$

For both experiments each evolution process (during which several populations are generated) proceeds until the `sigma`[2] value of a population is less than 0.01.

5.1 Experiment 1 Setup

The walking behaviors that have been hand-designed for SPOT only make use of the third shoulder (joint number 3 in figure 1) and the knee joint (joint number 4 in figure 1). During the execution of the walking behavior, the foot is aligned parallel to the body of the robot by a more or less passive control. The upper two joints of the shoulder (joint numbers 1 and 2 in figure 1) were not used and were held constant to minimize the complexity of designing the walking patterns for a human developer. This setup is also an appropiate base for the first experiment and allows a practical comparison between the human designed and the learned walking behaviors. Therefore, only the patterns of the third shoulder and the knee joint are learned in this experiment. In this case, the learning algorithm

[2] `Sigma` is an internal value of CMAES which is equivalent to the convergence of an evolution process.

would normally need to optimize a total of eight joint angle patterns, two per leg. However, after taking into account the symmetry constraints, the learning algorithm must only optimize the patterns of four joints (front-shoulder, front-knee, rear-shoulder, rear-knee) to learn a walking behavior. These four patterns result in a 16-dimensional object variable vector used by the learning algorithm.

The other parameters are held constant in order to minimize the dimensionality of the learning problem. The step period T is set to 3 seconds, and Δt_{sw} is set to 0.4 seconds. The phase shifts are chosen such that the robot will execute one swing movement after another, starting with the rear right leg, followed by the front left, the rear left, and finally the front right leg. These empirical values are based on experience gained from manually designed walking patterns of the robot. The step period of three seconds means about three steps per leg during the evaluation time of ten seconds.

5.2 Experiment 2 Setup

In this experiment, the evolutionary algorithm is forced to manage *all* degrees of freedom of the legs. Making use of all joints for the learning algorithm increases the possible space of solutions considerably. First tests of various fitness functions showed that many solutions of this space performed well in the simulation but would not be transferable to the real SPOT system. This is because the simulated robot possessed certain physical possibilities which would cause the real robot to be damaged (e. g. the way the robot could contact the ground). Upgrading the simulation to model these material weaknesses was not realizable in this study. Instead, the last two criteria of the fitness function are used to shape the evolutionary process to find a transferable behavior. In this experiment the algorithm has to learn a total of 12 joint patterns (6 for the front legs, and 6 for the rear legs), resulting in 48 parameters. The step period T and the swing period Δt_{sw} are set to the same values as in the first experiment. In contrast to the first experiment, the phase shifts are also learned by the evolutionary algorithm, beginning with the values used in the first experimental setup.

5.3 Results

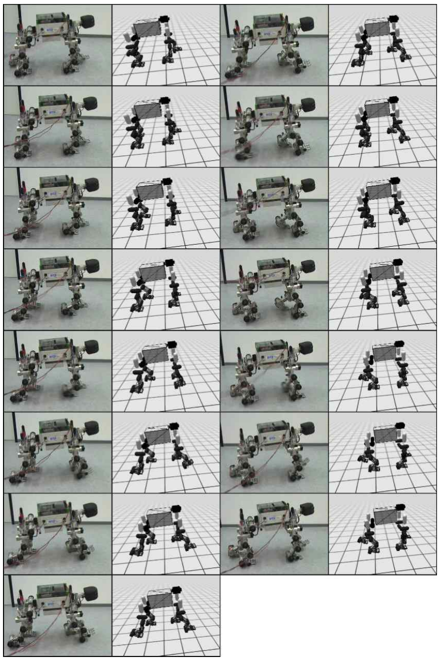
All learned behaviors performed very well in the simulation and the fitness criteria forced the evolutionary process to produce several results that were transferable. The non-transferable results typically made too much use of the toes and their joints, which are not as stable on the real robot as on the simulated one. Table 1 shows some important variables that were calculated over 50 independent runs of both experiments. The standard deviations of most variables shown in Table 1 are very small, and thus the values generated by the fitness function also have a low standard deviation.

The second experiment resulted in significantly[3] better fitness values than the first experiment ($p < 5 \times 10^{-8}$). These better fitness values were primarily due to the fact that the robot reached a significantly higher speed (i. e. a further distance was travelled during a fixed period of time) ($p < 2 \times 10^{-7}$).

[3] Significance levels were computed using a one-sided Student's t-test.

Table 1. Statistics of various important variables over 50 independent evolution processes of both experiments

Variable	Optimum		Average		Std. Deviation	
	Expt1	Expt2	Expt1	Expt2	Expt1	Expt2
Fitness	4.98	4.28	8.66	6.43	1.60	1.26
Num. of Evaluations	2292.00	7590.00	4187.52	11838.60	1020.86	1423.78
Distance in m	3.17	4.05	2.10	2.84	0.59	0.35
Avg. Height in m	0.59	0.61	0.56	0.54	0.02	0.04
Avg. Torque in N-m	2.64	2.71	3.61	4.03	0.30	0.49
Avg. Load in N-m	10.32	10.62	16.93	15.79	2.47	2.56
Avg. Pitch in degrees	0.74	0.86	2.86	6.60	1.02	4.51
Avg. Roll in degrees	0.52	0.10	2.86	4.28	1.00	2.19



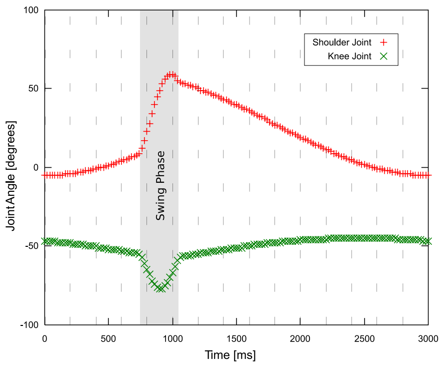


Fig. 4. A time-progression of snapshots showing a learned walking behavior that was executed on both the simulated and real robot. The images are arranged in an chronological order from top to bottom, starting with the left column. The curves that are executed by the front legs are shown in the plot on the right. These angle-values were recorded on the real robot during the execution of the evolved walking patterns. The dotted vertical lines indicate the points at which the snapshots were taken.

Some walking behaviors that had a good fitness in both experiments were transfered directly and without modifications to the real robot. One transfered walking behavior is shown in figure 4. When executed on the real robot, it proved to be stable, closely approximating the behavior of the simulated robot. The plot in Figure 4 shows the curves for the shoulder and the knee joint of the front right leg. The dotted vertical lines indicate the exact moments at which the pictures in this figure were taken. It can be seen from this image sequence that the real robot's behavior is very similar to the behavior of the simulated robot. As compared to the manually designed gaits, the learned gaits results in a faster and more balanced dynamic walking behavior. The main difference between most of the gaits learned and the manually designed behavior lies in that during particular stages of the walking process of the learned gaits two legs are in the air at the same time, while in the manually designed behavior only one leg is in the air.

6 Conclusions and Future Work

The results of the experiments presented are promising. Not only were walking behaviors learned in a simulation, but some of the learned gaits were successfully transferred to the real robot and resulted in a stable walking behavior. Additionally, by using an evolutionary algorithm, the dynamic properties of the robot were taken advantage of, yielding results that improved upon the previous walking patterns that had been manually designed by our research group. We believe that this approach can be transferred onto most other existing legged robot platforms. However, many of the learned walking patterns (mostly from the second experiment) are otimized highly to the simulation features. Therefore they do not produce a stable walking behavior on the real robot. A task for the future will be to change the experiment's setup in order to produce more robust walking patterns. Within that task we will analyse the differences between the simulated and the real robot's behavior and formalise the transferablitity of walking patterns. Another possible area of future work could involve applying the optimization methods presented here to a more general reflex walking model. In addition, the remaining designer bias might be reduced by replacing the model of distorted sine waves for describing the joint movements by a more general model like a recurrent neural network which can incorporate sensor information more easily.

References

1. Beer, R.D., Gallagher, J.C.: Evolving dynamical neural networks for adaptive behavior. Adaptive Behavior 1(1), 91–122 (1992)
2. Beer, R.D., Quinn, R.D., Chiel, H.J., Ritzmann, R.E.: Biologically inspired approaches to robotics: what can we learn from insects? Communications of the ACM 40(3), 30–38 (1997)
3. Gomi, T., Ide, K.: Emergence of gait of a legged robot by collaboration through evolution. In: IEEE World Congress on Computational Intelligence (WCCI 1998), New York. IEEE Press, Los Alamitos (1998)
4. Grillner, S.: Neural networks for vertebrate locomotion. Scientific American 274(1), 64–69 (1996)

5. Gruau, F., Quatramaran, K.: Cellular encoding for interactive evolutionary robotics. Technical report, Amsterdam, The Netherlands (1996)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
7. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous evolution of gaits with the sony quadruped robot. In: Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, 13-17, 1999, vol. 2, pp. 1297–1304. Morgan Kaufmann, San Francisco (1999)
8. Ito, K., Matsuno, F.: A study of reinforcement learning for the robot with many degrees of freedom - acquisition of locomotion patterns for multi-legged robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 3392–3397 (2002)
9. Kassahun, Y., Edgington, M., de Gea, J., Kirchner, F.: Exploiting sensorimotor coordination for learning to recognize objects. In: Twentieth International Joint Conference On Artificial Intelligence (IJCAI 2007), Hyderabad, India, pp. 883–888 (January 2007)
10. Lewis, M.A., Fagg, A.H., Solidum, A.: Genetic programming approach to the construction of a neural network for control of a walking robot. In: Proceedings of the International Conference on Robotics and Automation, Nice, France, 12-14 1992, vol. 3, pp. 2618–2623. IEEE Computer Society Press, Los Alamitos (1992)
11. Reil, T., Husbands, P.: Evolution of central pattern generators for bipedal walking in a real-time physics environment. IEEE Trans. Evolutionary Computation 6(2), 159–168 (2002)
12. Röfer, T.: Evolutionary gait-optimization using a fitness function based on proprioception. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 310–322. Springer, Heidelberg (2005)
13. Schwefel, H.-P.P.: Evolution and Optimum Seeking: The Sixth Generation. John Wiley & Sons, Inc., New York (1993)
14. Smith, R.: Open dynamics engine (2005), `http://www.ode.org`
15. Spenneberg, D.: Bioinspirierte Kontrolle von Laufrobotern. PhD thesis, Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany (July 2006)
16. Spenneberg, D., Albrecht, M., Backhaus, T., Hilljegerdes, J., Kirchner, F., Strack, A., Zschenker, H.: Aramies: A four-legged climbing and walking robot. In: Proceedings of 8th International Symposium iSAIRAS, Munich (2005)
17. Zhang, J., Chen, Q.: Learning based gaits evolution for an AIBO dog. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1523–1526 (September 2007)

Driving Cars by Means of Genetic Algorithms

Yago Saez[1], Diego Perez[1], Oscar Sanjuan[2], and Pedro Isasi[1]

[1] Carlos III University, Madrid, 28911, Spain
[2] Oviedo University, Oviedo, 33002, Spain

Abstract. The techniques and the technologies supporting Automatic Vehicle Guidance are an important issue. Automobile manufacturers view automatic driving as a very interesting product with motivating key features which allow improvement of the safety of the car, reducing emission or fuel consumption or optimizing driver comfort during long journeys. Car racing is an active research field where new advances in aerodynamics, consumption and engine power are critical each season. Our proposal is to research how evolutionary computation techniques can help in this field. As a first goal we want to automatically learn to drive, by means of genetic algorithms, optimizing lap times while driving through three different circuits.

1 Introduction

Automatic Vehicle Guidance (AVG) has been addressed by numerous researchers, i.e. [12], [15], [17], [13], [16], [20], as an engineering problem through different approaches, and the most promising ones are being engineered on real prototypes, i.e., [13], [5], the Buick from the California PATH project or [7], or the cars which participate each year in the annual competition organized by the Defense Advanced Research Projects Agency (DARPA) for driverless cars, [6]. The study of the suspension system has been an interesting topic for researchers because it contributes to the car's handling and braking for good active safety and driving pleasure, and keeps vehicle occupants comfortable and reasonably well isolated from road noise, bumps, and vibrations. This parameter optimization has been done with Genetic Algorithm (GA) with promising results in several works, such as, [8], [9] and [24]. One of the first works on this topic was carried out at the Carnegie Mellon University by Sukthankar et al. in 1996, [16]. This work uses reasoning modules which combine high level task goals with low-level sensor constraints, proposing a Population Based Incremental Learning (PBIL) [2] for an automatic setting of the module parameters. For the simulation the system uses a program called SHIVA (Simulated Highways for Intelligent Vehicle Algorithms) which reproduces a micro-simulation of vehicles moving and interacting in a user-defined roadway. Two years later the PBIL was compared to the GA in this same framework, [3]. The results of these works were quite motivating since they showed the potential for intelligent behavior in tactical driving. Other interesting work carried out in 1996 by Pyeatt et al. from Colorado University deals with simulated race car driving [14]. In this case a study about autonomous driving was developed based upon RARS simulator software (the one which has influenced TORCS). The results showed that the RARS simulator was adequate for developing the test framework and that neural networks can be competitive techniques for producing autonomous racing cars.

In 1998, Bernard et al.[4] from Iowa State University illustrated the power of GAs to model driver/vehicle behavior. In fact, their work determined how fast and safe a given vehicle model could be. The experimental framework was to drive through a short course without hitting a cone or lifting its wheels. In 2004, Floreano et al., [10], used a GA for tuning up a neural network which visually recognizes edges, corners and height, resembling strategies observed in simple insects which obtained results that performed equal or better than well trained human drivers tested on the same circuits. In 2005, Sun et al. [18], used a GA for optimizing the parameters of a set of Gabor filters in the context of vehicle detection from images. They successfully tested the proposed framework on real data and improved the performance of on-road vehicle detection. In the same year another interesting approach was proposed for the automated evolutionary design of driving agents, [11], [19]. This work showed how GAs can help in the task of designing an agent able to remotely operate a scale racing car. This work revealed that on long runs the agent's operated car was 5% slower than the human operated one. Working with the evolving weights of a neural network, Julian Togelius et al. compared, with their own simulation, simulated cars with evolved neural network controllers (in first-person and third-person) [23], [22]. They extended their work to a more complex case of two cars competing against each other in the same track at the same time, [21], using evolutionary strategies to solve the problem of the coevolution. Finally, an interesting study which compares neuroevolution and genetic programming in the same environment can be found in [1].

2 Problem Analysis and Design

As can be seen in the previous section, the AVG is a very difficult and widespread problem. It has been tackled from different points of view, involving, among others, robotics, artificial intelligence, computer engineering, telecommunications, signal and image processing, or control and automation techniques. Our proposal in this work is to divide the problem of the AVG starting from the optimization of a lap trajectory in a known circuit. The idea is to increase the difficulty of this scenario in future developments in order to solve the AVG problem. Thus, in this paper, we have dealt with the problem of minimizing a single lap time using Evolutionary Computation (EC) techniques, and without taking into account obstacles or consumption factors. This work will be used as a base for hereafter extensions of the goal. As we wish to reproduce human behaviour, we have designed a codification based on actions that are applicable every certain number of meters. This constant number of meters divides the circuit into segments. The actions that can be done in each segment are acceleration (value between 0 and 1, where 0 means no pressure and 1 total force) and steering (measured with a real number between -1 and 1, where negatives values mean left turns and 0 means no turn at all). This codification encodes the individuals using 2 chromosomes: one for the acceleration value and the other for the steering. Each chromosome has the same number of genes as segments. The genes contain real values. It must be taken into account that the segment size needs to be adjusted because long segment sizes will cause shorter individuals (which is ideal for the GA) but it will offer less precision to the driver. Therefore, a trade off must be found in order to obtain a good base individual. One of the biggest problems faced when trying to optimize a car trajectory in a circuit by means of GAs is how to choose the first configuration used. This is because traditionally GA approaches evolve directly from an initially random generated configuration. As it is very difficult to obtain automatic driving evolution from individuals that have been generated by chance, we decided to develop a traditional algorithm to obtain a base trajectory which can complete a lap, although finishing it in poor times. The mechanism of this algorithm is simple, it tries, for each segment, all possible combinations of steering and acceleration (using, only for this base individual, a discretization of the allowed values with a granularity of 0,1), looking for the pair of values which maintains the vehicle closer to the center of the track. Once all segments have been evaluated we have obtained the individual taken as a basis for the GA, then the driving learning process can start. The operators used in the GA are the following:

1. Selection: a size three tournament has been implemented in all experiments.
2. Mutation: the mutation operator used is executed over each one of the genes of the final individual. Under certain mutation probability, each gene can be altered by chance so the new value will be a quantity not too far from the original.
3. Crossover: two crossover types were used in this project:
   (a) Uniform: each one of the genes of the final individual has equal probability of coming from each one of its parents.
   (b) By stretches: the goal of this approach is to test if a group of genes can encode a good way to, for instance, take a turn. In this case, the probability that determines the source of each gene is the same, but now these genes are exchanged in groups, instead of one by one.
4. Elitism: elitism of one individual has been used in all experiments.

The simulator used in this project to test our individuals is TORCS (The Open Racing Car Simulator), chosen because it is open source and gives us the flexibility to modify and manage the process internally. The car used in all the test is Nascar RWD (Nascar category), taking its fuel, aerodynamics and suspension configuration by default. It is also important to note that the lap time stored is from only the first lap. After the first lap the race is restarted in order to evaluate the next individual. All the experiments were conducted in three different circuits of diverse complexity, as can be seen in Figure 1:

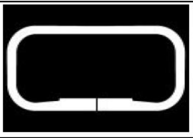
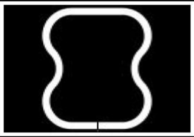
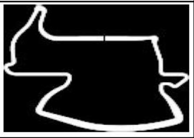
		
A-Speedway	**E-Track 5**	**Aalborg**
1908,32 m Length	1.621,73 m Length	2587.54 m Length
25 m Width	20 m Width	10 m Width

Fig. 1. Circuits tested

2.1 Results with Fixed Segment Size (IT1)

In the A-Speedway circuit, the experiments were divided into 2 groups, depending on both types of crossover techniques explained before. Each experiment was executed through $1,000$ generations and the segment size was fixed to 10 meters. Five experiments of each set were executed. As can be seen in Table 1, results with both crossover strategies are very similar, but the uniform case provides more stability in them. All experiments ended with lap times under 40 seconds, which is a very big improvement compared with the initial 76 seconds of the individual of reference. Most of the experiments finished with a very good tendency, and it is expected that the times keep on improving during further generations (only $1,000$ were conducted). After the good results obtained in the first circuit, we tried another set of experiments on a a bit more complicated track: the E-Track5. The results were good, but the improvements found by the GA on this track (10.61 seconds on average, 14.45%) were not so good as the ones obtained for the A-Speedway (39.69 seconds on average, 50.89%). To understand the problem we observed the behaviour of the generated drivers and we concluded that the size of the segment in circuits with more turns affects the performance of the driver, because some parts of the circuit need more precision driving than others. To test the influence of this parameter we ran 2 different sets of experiments using different constant segment sizes.

As can be seen in Table 1, the results of the experiments with longer segments are quite worse than the ones with shorter stretches. In the first case, the vehicles do not have enough precision to drive through the circuit because the size of the segment. Therefore, it is complicated to obtain a good performance in the lap with large segments, even for the individual of reference. In contrast, shorter segments allow drivers to have more precision when driving through the track. Finally, we decided to see what happens in the Aalborg track. This circuit

Table 1. Results in A-Speedway and E-Track 5

Circuit	Pop. size	Mutation	Crossover	Segments	Base ind. time	GA time(mean)	σ
A-SpeedWay	100	0.005-0.01	uniform	191	76.064 secs	36.479 secs	2.99
A-SpeedWay	100	0.005-0.01	stretches	191	76.064 secs	36.369 secs	4.51
E-Track 5	100	0.005-0.01	uniform	163	129.992 secs	112.468 secs	17.871
E-Track 5	100	0.005-0.01	uniform	326	74.108 secs	63.493 secs	3.915

is the most difficult one of the three tracks tested: it is longer, it has more turns and it is much narrower than the others. The base individual has been generated using a segment size of five meters, in order to provide a high precision for the driving, and ends a lap in 288.044 seconds. The GA was configured using uniform crossover and, like the previous circuits, different combinations of mutation values were tested on several experiments. The best of them got 178.478 seconds, so a reduction of almost 2 minutes from the base time was obtained.

2.2 Evaluating the Results with Other Simulator Bots

However, the results presented so far, even reducing the base individual times, need to be validated against real drivers. As real drivers need to be trained, and can be biased, our proposal is to compare the results with the lap times performed by the simulator bots. The simulator provides several programmed bots that have been used to check whether our lap times are really competitive or not. Figure 2 shows the fitness curve of the best individuals of A-Speedway compared with simulator bots' lap times.

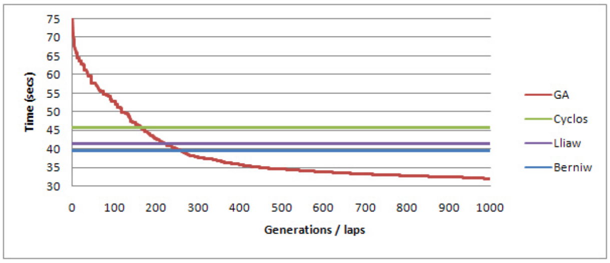


Fig. 2. A-Speedway best individual vs. bots (fitness curve)

In this figure can be clearly seen that the final lap time of our experiments is much better in A-Speedway than the best times performed by the bots. Another interesting comparison can be made by taking the lap times performed by human players, with nearly one hour of experience, in this circuit. The best human time obtained was 35.510 seconds (while ours is 31.962 seconds) so our GA individual is even better than a human driver. Analyzing E-Track 5, our lap times are worse than the bot ones, with a difference ranging between 12 and 19 seconds. As the difficulty grows, it becomes worse. For the Aalborg track, the simulator bots perform, depending on their skills, from 96.440 to 78.970 seconds, and the GA's best achieved result is 178.478 seconds. Allowing the GA to run during more generations would have decreased the lap time, but it is important to consider an improvement in the GA in order to evolve faster.

2.3 Variable Segment Size (IT2)

The results obtained in the first part of this research were promising but, in the most complex circuits, the lap time reduction could not reach the times performed by the simulator bots. In order to improve the results and take a step forward in the problem, some modifications were proposed:

- Variable sized segments: the most important modification implemented is the adaptation of the individual to variable segment sizes. In the case of real driving, the number of corrections of speed and steering depends greatly on the features of the track point where the car is located on. Indeed, a higher number of actions is required when the vehicle is in a bend. The point, then, is how to find a procedure to determine the size of each segment of a circuit. The idea is to create long segments for straight parts and short ones for turns, in order to provide the individual with more precision for the trajectory. Fortunately, the TORCS simulator provides us with a segment division that accomplishes this requirement, so they were used in the new codification. Nevertheless, if we did not have this information, we could obtain these segments analyzing the curve tangent of the track, considering more segments in sharp turns (higher variations of tangent value) and less in smooth stretches (lower variation).

- Changes on the individual of reference: because of the changes in segment sizes, a new problem arises when the base individual is searched again. With the new segment sizes, some of them are so short that the individual interprets them as straight segments. Then, the vehicle does not need to steer to keep centered in the track, and obtains higher speed turning with an unproper speed. Several solutions were considered, but in the end we defined a new parameter in charge of setting the maximum speed that the car was allowed to reach. Therefore, the new algorithm used to find the base individual needs only to find the best steering value to keep the car centered while keeping its velocity near (and always under) the speed limit. The application of this new algorithm brought us two main advantages: (1) the execution time spent was decreased dramatically because the search space is smaller and, (2) the lap time of the individual of reference was reduced. This improvement is an outcome of the changes made which provide more precision with the vehicle and, consequently, better lap times. Table 2 shows the new values for segments and their initial lap times.

Table 2. New parameters for the genetic algorithm

Circuit	Segments (1st Iteration)	Segments (2nd Iteration)	Base Time (1st Iteration)	Base Time (2nd Iteration)
A-Speedway	384	244	76.06 secs	60.17 secs
E-Track 5	326	334	74.67 secs	68.33 secs
Aalborg	518	371	288.04 secs	158.39 secs

It is remarkable how the new changes reduce the base times in all circuits, being the reduction in the last circuit very promising: 158.39 seconds. This lap time is much lower than the time obtained by the GA with the IT1 codification (178.478 seconds).

- Changes on the GA: a steady state has been implemented to provide more generations. It works creating a new individual in each generation, using the usual operators (tournament selection, uniform crossover and mutation, as were used before), and comparing it with the worst one of the population. In the case where the new individual is better than the old one, it is substituted and the algorithm continues creating a new individual.

Once the previous modifications were made, a test set was executed in order to check if better results could be obtained. All of them were executed in the third circuit (Aalborg). As Aalborg is the most complex of the circuits tried, we considered that good results on it would mean good results in the others (A-Speedway and E-Track 5). After several tests, the mutation value chosen was 0.005. All the experiments were executed through 100, 000 steady state generations. The results found are summarized in Table 3. As can be seen in this table, the final lap times are competitive, and even more if we compare them with the best times obtained in the previous iteration. The improvements in relation to IT1 are measured in a difference of almost 90 seconds on average.

Table 3. Summary of 40 experiments in Aalborg (IT2)

IT2	Generations	Base time	Reduction	Final Time
Mean	1,000	158.39 secs	43.10 secs	115.28 secs
Best result	1,000	158.39 secs	**59.79** secs	**98.59** secs

The improvements made in relation to IT1 are obviously due to the changes done to the base individual. However, the best lap times achieved by the GA in the IT2, are very close to the ones performed by the simulator bots. The final lap time stopped at 98.592 seconds, only 2 seconds away from the lap time of one of the bots. Attending to the mean, this approach is still 32.04 seconds (on average) away from the best TORCS simulator bot.

2.4 Variable Segment Size and Gene Value Discretization (IT3)

As the results obtained after the IT2 modifications did not reach the bots lap times, we made some additional changes to the algorithm, with the aim of performing competitive lap times. The changes made in this third iteration of the paper are the following:

Initial population mutation: in this iteration, the whole initial population is mutated before starting in order to have more diversity for the steady state algorithm. This change helps the algorithm to perform better by providing a fast fitness reduction during the first generations. It is important to note that, in this mutation, only one individual remains intact (as the individual of reference) to ensure having at least one which can complete a lap.

Discretization of genes values: from the beginning of the project, the values of the genes have been continuous (real value codification). However, we can consider the relevance of the values beyond the fourth or fifth floating point number insignificant. For instance, the difference between a turn value of 0.0001 and 0.0002 cannot be appreciated in the driving process and, what it is worst,

Table 4. Summary of experiments on Aalborg (IT3)

IT3	Generations	Base time	Reduction	Final Time
Mean	1,000	158.39 secs	65.05 secs	93.34 secs
Best result	1,000	158.39 secs	**73.81** secs	**84.57** secs

increases the search space, affecting negatively to the performance of the algorithm. In addition, a physical instrument for steering will probably not be able to work with such a precision. The problem of this approach is that, as mutation is a procedure that can change the value of a gene in any quantity, its final value will not be discrete and the search space becomes extremely large. Because of that, all the genes values have been discretized and their precision has been set to two decimal numbers.

Changes in mutation procedure: due to the previous modification, the mutation procedure has changed in order to fit in the new gene values. However, another important update has been performed in this process. All over the GA, some kind of cultural information is tracked: when an individual is mutated and the new lap time is better than the one of its parents, the values of these successful mutations are stored in a special individual. The information kept here is used as a reinforcement for the next mutation chances, performing the gene modification oriented to the successful mutations of its neighbors. This change is based on the fact that contiguous segments usually perform similar actions in this problem.

Once the changes were made, another test set was executed. The results, which are gathered in Table 4, are significantly better, obtaining competitive lap times on average, and with a best result of 84.578 seconds. This new algorithm obtains a time reduction of 22 seconds on average from the times achieved by the base individual, and its standard deviation is also lower. Comparing our times with the simulator bots, the results obtained are, on average, between the worst and the best simulator bot lap times, so the objective of achieving competitive lap times has been reached. The evolution of the best individual achieved by the IT3 GA compared to the bot lap times can be seen in Figure 3.

As can be seen, fitness value decreases smoothly from the base individual lap time, achieving competitive values before the half point of the GA execution.

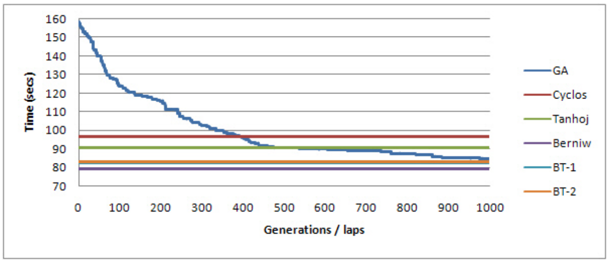


Fig. 3. Aalborg best individual vs. simulator bots (fitness curve)

This behaviour is common in most of the experiments of this test set. Furthermore, in the last generations, the decreasing of the lap time continues, which indicates that the results will be even better with longer executions. Moreover, results on average also improve part of the times obtained by the simulator bots.

3 Conclusions and Future Work

The use of GAs has been proved to be a successful technique for the optimization of a lap trajectory in a known circuit. In the first experiments performed in this paper, the results in complex circuits showed us that we were very far away from competitive lap times. However, by analyzing and studying the algorithm, some changes were made to the GA. With these changes the GA became good enough to obtain lap times comparable with the ones achieved with the TORCS simulator bots. This was tested by means of experiment sets with a significant number of runs. The relevance of these successful results is not the fact of beating the bots of the simulator. The relevance is the discovery, in the process of doing so, of each one of the features which make this problem difficult and interesting to solve. One of the problems of this approach is that the trajectory obtained in the evolving process depends highly on the initial state of the car. If noise in sensor readings or a change in the initial position happens, the trajectory evolved would not be valid. The same happens if we need to perform more than one lap. Avoiding these problems is one of the main objectives for future developments. Two branches of future work can be the basis of our new research lines. On one hand, applying different techniques, such as evolutionary strategies and/or multi-objective optimization. These techniques can be used to find out new possibilities for solving this problem in order to obtain better lap times than any bot or human player. On the other hand, we can add complexity to this challenge; for instance, we can try to include fuel consumption, gear changing, overtaking other vehicles and, why not, a whole race planning.

References

1. Agapitos, A., Togelius, J., Lucas, S.: Evolving controllers for simulated car racing using object oriented genetic programming. In: Procs. Conference on Genetic and Evolutionary Computation, pp. 1543–1550. ACM, New York (2007)
2. Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: Prieditis, A., Russel, S. (eds.) The Int. Conf. on Machine Learning 1995, San Mateo, CA, pp. 38–46. Morgan Kaufmann Publishers, San Francisco (1995)
3. Baluja, S., Sukthankar, R., Hancock, J.: Prototyping intelligent vehicle modules using evolutionary algorithms (1998)
4. Bernard, J., Gruening, J., Hoffmeister, K.: Evaluation of vehicle/driver performance using genetic algorithms, society of automotive engineers
5. Bertozzi, M., Broggi, A., Conte, G., Fascioli, A.: The experience of the argo autonomous vehicle. In: Procs. Spie 1998 (1998)
6. Chen, Q., Ozguner, U., Redmill, K.: Ohio state university at the 2004 darpa grand challenge: Developing a completely autonomous vehicle. IEEE Intelligent Systems 19(5), 8–11 (2004)

7. Collado., J.M., Hilario, C., de la Escalera, A., Armingol, J.M.: Self-calibration of an on-board stereo-vision system for driver assistance systems. In: Intelligent Vehicles Symposium, pp. 156–162 (2006)
8. Cambiaghi, D., Vetturi, D., Gadola, M., Manzo, L.: Semi-active strategies for racing car suspension control, paper no. 962553, university of brescia
9. Hacker, K., Kasprzak, E.M., Lewis, K.: Exploring the design tradeoffs and computational savings of executing vehicle simulations in a parallel computing environment. In: ASME Design Automation Conference (2000)
10. Floreano, D., Kato, T., Marocco, D., Sauser, E.: Coevolution of active vision and feature selection. Biological Cybernetics 90, 218–228 (2004)
11. Hemmi, H., Tanev, I., Joachimczak, M., Shimohara, K.: Evolution of the driving styles of anticipatory agent remotely operating a scaled model of racing car. In: Procs. of the IEEE Congress on Evolutionary Computation, Edinburgh, 2-4 September 2005, vol. 2, pp. 1891–1898. IEEE, Los Alamitos (2005)
12. Niehaus, A., Stengel, R.F.: Probability-based decision making for automated highway driving. In: Vehicle Navigation and Information Systems Conference, vol. 2, pp. 1125–1136 (1991)
13. Pomerleau, D., Jochem, T.: Rapidly adapting machine vision for automated vehicle steering. IEEE Expert: Special Issue on Intelligent System and their Applications 11(2), 19–27 (1996); see also IEEE Intelligent Systems
14. Pyeatt, L., Howe, A., Anderson, C.: Learning coordinated behaviors for control of a simulated robot (1996)
15. Siegle, G., Geisler, J., Laubenstein, F., Nagel, H.-H., Struck, G.: Autonomous driving on a road network. In: Procs. of the Intelligent Vehicles 1992 Symposium, Res. Center Commun., Robert Bosch GmbH, Hildesheim, pp. 403–408 (1992)
16. Sukthankar, R., Baluja, S., Hancock, J., Pomerleau, D., Thorpe, C.: Adaptive intelligent vehicle modules for tactical driving (1996)
17. Sukthankar, R., Pomerleau, D., Thorpe, C.: Shiva: Simulated highways for intelligent vehicle algorithms (1995)
18. Sun, Z., Bebis, G., Miller, R.: On-road vehicle detection using evolutionary gabor filter optimization (2005)
19. Tanev, I., Joachimczak, M., Shimohara, K.: Evolution of driving agent, remotely operating a scale model of a car with obstacle avoidance capabilities. In: Procs. of the Conference on Genetic and Evolutionary Computation, pp. 1785–1792. ACM, New York (2006)
20. Thorpe, C., Jochem, T., Pomerleau, D.: Automated highway and the free agent demonstration (1998)
21. Togelius, J., Lucas, S.: Arms races and car races. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 613–622. Springer, Heidelberg (2006)
22. Togelius, J., Lucas, S.M.: Evolving robust and specialized car racing skills. In: Yen, G.G., et al. (eds.) Procs. IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16-21, 2006, pp. 1187–1194. IEEE Press, Los Alamitos (2006)
23. Togelius, J., Simon, M.: Evolving controllers for simulated car racing. In: Procs. IEEE Congress on Evolutionary Computation (2005)
24. Wloch, K., Bentley, P.J.: Optimising the performance of a formula one car using a genetic algorithm. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 702–711. Springer, Heidelberg (2004)

AGE-P: A Platform for Open Evolution

Ralf Salomon and Stefan Goldmann

Faculty of Computer Science and Electrical Engineering
University of Rostock, 18051 Rostock, Germany
{ralf.salomon,stefan.goldmann}@uni-rostock.de

Abstract. Smart-appliances ensembles are highly dynamic device collections in which devices can leave and join at any time without notice. Due to the high system dynamics, such ensembles cannot employ standard evolutionary algorithms for their internal self-organization processes. Therefore, this paper proposes a new evolutionary framework, called the appliances-go-evolution platform (AGE-P). The simulation experiments indicate that AGE-P is able to properly cope with the peculiarities of smart-appliances ensembles and that it is thus a suitable option for their self-organization processes.

1 Introduction

Evolutionary algorithms of various sorts solve technical problems by utilizing some selected concepts from natural evolution [2,5,6,9,12]. They describe a technical (optimization) problem as a set of n problem-specific parameters x_i, also called genes. The set of genes is called a genome and is tightly embedded into an object, called individual. Typically, an evolutionary algorithm applies its random variation operators, such as mutation and recombination, to an individuals' genes. As a consequence, these random variations change the individuals' fitness values. A subsequent selection process exploits these fitness variations in order to gain some progress. It should be obvious that both the fitness evaluation and the selection process consider the genomes as atomic entities. In summary, the notion of an *individual* as the container of its genome is a very fundamental concept in all evolutionary algorithms.

The concepts described above are quite generic by their very nature. The pertinent literature on evolutionary algorithms presents a huge number of successful applications that can be found in areas as diverse as machine learning, combinatorial problems, VLSI design, breast cancer detection, evolutionary robotics, and numerical optimization in general. But in its canonical form, the concept of an individual is not suitable for all types of applications. For example, when evolving structures, such as the topology of a neural network, the number n of parameters x_i is generally not not known in advance. Rather, the number n of parameters itself is the result of the actual evolutionary process. As a relief, previous research has developed the concept of variable-length genomes [7,8,11]. This option allows an individual to grow and shrink its genome, and thus to adapt to changing demands. But still, with its genome, an individual constitutes a solid, atomic, and monolithic entity, which is fundamental to all evolutionary algorithm.

Even with the concept of variable-length genomes, evolutionary algorithms cannot be directly utilized in all application domains. Section 2 briefly describes an example application called smart-appliances ensembles [1,10,13]. The term "smart-appliances ensemble" refers to everyday-life devices that are equipped with some computational resources and that are supposed to self-organize according to the users' needs. The following properties are closely linked to smart-appliances ensembles: (1) they are dynamic by their very nature in that devices may join or leave the ensemble at any time without notice; (2) the physical properties of every device are known only to itself and *not* the rest of the system; and (3) a smart-appliances ensemble should not induce any user-based modeling and/or administration; rather, devices might be freely added and be freely removed, which also includes device failures.

The discussion presented above suggests that the conventual usage of individuals and genes does not match the dynamic and modeling-free nature of smart-appliances ensembles. Therefore, Section 3 proposes a new evolutionary framework, called the appliances-go-evolution platform (AGE-P). A key feature of AGE-P is that it physically distributes the genome as well as the mutation operators across all the appliances. This way, the genome grows and shrinks as devices come and go, and thus is naturally adapting to the ensemble dynamics.

For validation purposes, Section 4 describes the office lighting problem in which several light sources are distributed within a typical office space. The light sources are supposed to autonomously dim such that all users have the specified illuminations at their desks. In this educational example, neither the number of light sources nor their physical properties are known to the system. Rather, all light sources randomly change their activation, and the resulting effects are subsequently fed back by the sensors.

The results, as presented in Section 5, indicate that the proposed AGE-P approach is able to solve the office lighting problem and that it is able to cope with all the mentioned system dynamics. Finally, Section 6 concludes this paper with a brief discussion.

2 Problem Description: Smart-Appliances Ensembles

Smart appliances refer to devices, such as laptops, personal digital assistants, cellular phones, beamers, window blinds, light bulbs, and the like, that are present in everyday life and that are accessible via some electronic interfaces. Smart-appliances are considered an *ensemble* if they coherently work together such that they support their users in an autonomous and non-invasive way [1,10,13]. In order to reach this goal, the devices should employ a self-organization process. The self-organization process itself requires access to a proper communication infrastructure, such as a wireless network and Bluetooth, by means of which the devices exchange messages with each other.

The smart lecture room is a good educational example. It may consist of a certain number of laptops, a few beamers, some light sources, and window blinds. Suppose that a user is about to do a presentation and that the room is illuminated too much. Then, the ensemble is supposed to properly dim the lights and/or close the window blinds.

A common approach for solving this task is to utilize rule based methods, such as ontologies [3,4]. In this modeling approach, the devices use rules to negotiate how to react to a given situation. However, the utility of such rule-based modeling is very limited in the problem at hand. Every joining or leaving device would impose some administrative work, which is but desirable. Rather, the ensemble is supposed to self-organize in order to work for the users and not vice versa. Thus, a modeling approach would not be the method of choice. As has already been discussed in the introduction, even standard evolutionary algorithms are not suitable for this application: the notion of an individual as the container of its genome is highly useless, since devices and thus genes come and go *without* notice.

3 The Appliances-Go-Evolution Platform

This section proposes a new evolutionary scheme called *the appliances-go-evolution platform*, or AGE-P for short. AGE-P abandons any central processing; rather, it *physically* distributes all the operations as well as data structures involved across all actuators and sensors. This means, for example, that in AGE-P, every gene only resides in the device to which it belongs. Consequently, every device hosts its own mutation operator, which it applies only to its private gene. In other words, none of the other components has any knowledge about a device's gene value or its particular variation operator. A consequence of the chosen approach is that removing or adding devices automatically removes or adds the associated genes from the genome. Similarly, AGE-P distributes the fitness evaluation across the sensors present in a scenario. With these conceptional modifications in mind, AGE-P assumes the following setup:

1. All devices are split into two classes, sensors s_i and actuators a_i. Actuators are those devices that influence principal modalities, such as brightness, sound volume, and the like. Sensors, on the other hand, measure modalities.
2. Every sensor s_i is tagged with a target value s_i^t [1]. The overall goal of the AGE-P system is that all the differences $d_i = s_i - s_i^t$ vanish. In order to reach this goal, the sensors periodically communicate their differences d_i, which constitute the partial fitness contribution $f_i = (s_i - s_i^t)^2$, to all devices.
3. Every actuator a_i hosts its current activation, also denoted by a_i, as well as its private mutation operator. In order to perform an evolutionary process, every actuator also hosts its previous values along with their associated fitness values. In its most simple form, every actuator hosts one parent as well as one offspring gene value. Then, all actuators perform a (1+1) selection scheme, which indicates that the new parent gene is selected from

[1] It is generally assumed that the target values originate from either higher abstraction levels, such as an intention module, or given user settings. The discussion of the intension module, which is part of the ensemble's infrastructure, is beyond the scope of this paper.

the union of the previous parent and the offspring. This form is denoted as (1+1)-AGE-P for short.

4. AGE-P assumes that a propper communication infrastructure, such as Bluetooth, WLAN, DECT, or the like, is readily present.

In this physical setup, AGE-P works as follows:

1. Initially, all sensors are tagged with reasonable target values s_i^t. Similarly, all actuators choose suitable activations a_i. In case of light sources, these values might correspond to zero illumination.
2. Periodically, all sensors determine their current sensor values s_i, and broadcast the differences $d_i = s_i - s_i^t$.
3. On the arrival of new differences d_i, all actuators select their gene values from previous cycles, and apply their private mutation operators, i.e., $a_i \leftarrow a_i + m_i$, with m_i denoting the mutation operator of the ith actuator. As a consequence, the actuators' activations (randomly) change. The physics mediate these changes, which the sensors s_i feed back in the next cycle.
4. The process continues with step 2.

Dynamic changes, such as changing sun shine, broken actuators, changed locations of the actuators and/or the sensors, new actuators, etc., might invalidate the previously collected gene-fitness value pairs. To cope with these changes, AGE-P periodically skips the selection process and re-evaluates the so far best value. This re-evaluation might be triggered by any of the sensors or actuators.

4 Methods

In order to validate the concepts of the proposed AGE-P algorithm, this paper is using a simulation. Such a simulation-based approach considers only those aspects, which are technically relevant for the algorithm. For the sake of simplicity, the simulation models the illumination of a certain number of desks by a certain number of light sources. The remainder of this section presents a description of the considered scenarios as well as the used parameter settings.

Configuration of AGE-P. All experiments have been done with (1+1)-AGE-P. This notation indicates that the algorithm generates one offspring from one parent and that it selects the better one as the parent for the next generation.

Sensors s_i. AGE-P employs a user-specified number m of sensors s_i. In the validation study presented in Section 5, these sensors measure the illumination at various locations, e.g., the users' desks.

Fitness function f. All sensors calculate the difference $d_i = s_i - s_i^t$ as their partial fitness contribution. By means of a global communication infrastructure, all sensors broadcast their differences d_i across the system such that every device can calculate the ensemble's total fitness as

$$f = \sum_{i=1}^{m} f_i = \sum_{i=1}^{m} {d_i}^2 = \sum_{i=1}^{m} (s_i - s_i^t)^2 \ . \tag{1}$$

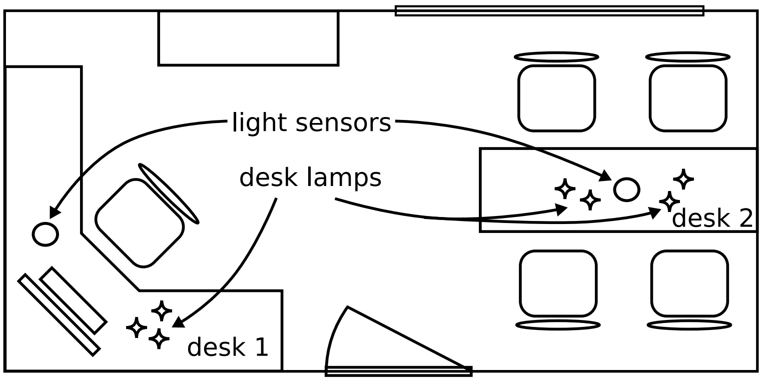


Fig. 1. The test scenario consists of some light sources as well as two desks with light sensors mounted close to the keyboards

Actuators a_i. The simulation employs n actuators a_i, which represent n light sources that are distributed in the environment. Every actuator employs its private mutation operator $a_i \leftarrow a_i + \sigma \cdot N(0,1)$, with σ denoting a private step size. Without loss of generality, all actuator values a_i are bound to $0 \leq a_i$. Unless otherwise stated, the step size is set to $\sigma = 0.1$ in all simulation scenarios. Please remember that these actuators are not explicitly known to the sensors, the fitness evaluation, or the system in general. Rather, the environment, i.e., the physics, autonomously mediate their modalities towards the sensors s_i.

Simulation setup. The simulation setup resembles a typical workplace situation as shown in Fig. 1. Such a room has several light sources a_i, desks, and light sensors s_i. The effect of light source a_i on sensor s_j can be modeled by a weight w_{ij}. The weights w_{ij} subsume all the relevant physical effects, such as the light sources' positions, their brightness, their illumination characteristics, etc. In addition, most rooms have one or several windows through which the sun might contribute some global illumination g. In mathematical terms, this paper utilizes the following (simplified) physical model:

$$s_j = g + \sum_{i=1}^{n} w_{ij} \cdot a_i \ , \qquad (2)$$

with w_{ij} set to values plausible for the scenario depicted in Fig. 1. It might be mentioned again, that the mathematical model of Eq. (2) is *not* part of AGE-P, but solely used for validation purposes within the simulation setup.

Depending on the chosen scenario (please, see below), the weights, the sensor target values, and the number of actuators spontaneously change over time, i.e., $w_{ij}(t)$, $g(t)$, $s_i^t(t)$, and $n(t)$ are time dependent. For the purpose of readability, the time t is omitted.

Scenario 1, System Startup. At startup time, all actuator values are set to $a_i = 0$ and $g = 0$, and the sensor target values are set to $s_1^t = 0.7$ and $s_2^t = 1.2$.

The ensemble is then responsible to power up the lamps to the desired level. This situation resembles an early winter morning, when the users enter the dark office.

Scenario 2, Scalability. This scenario increases the number of actuators from $n = 2$ to $n = 100$. The goal of this scenario is to test the scaling behavior of AGE-P.

Scenario 3, System Dynamics. This scenario focuses on the ensemble behavior in more dynamic setups in which light sources might fail or join. It starts off with two light sources per desk, i.e., $n = 4$. In simulation step $t = 150$, a light source of each desk fails. Then, in simulation step $t = 300$, the sensor target value of desk 2 is increased from $s_2^t = 1.2$ to $s_2^t = 1.5$. This might model a situation in which a different person starts working and might prefer a brighter desk. This situation implies that the illumination of the other desk should remain unchanged.

Scenario 4, External Effects. This scenario starts off like the first one. But in simulation step $t = 150$, the external (sunshine) illumination is set to $g = 1$, and is reduced to $g = 0.5$ im simulation step $t = 300$. This scenario models the influence of external modalities, which are outside the control of the ensemble. A further challenge is that during time steps $150 \leq t \leq 300$, the ensemble cannot reach the specified target values, since the external illumination already exceeds target value s_1^t.

5 Results

The simulation results of the four experiments are summarized in Figs. 2 to 5. On the x-axis, the figures show the simulation time, and on the y-axis, they show the target sensor values s_i^t, their actual readings s_i, and the global ensemble fitness f (Eq. (1)). All data were obtained from 500 independent runs. From all

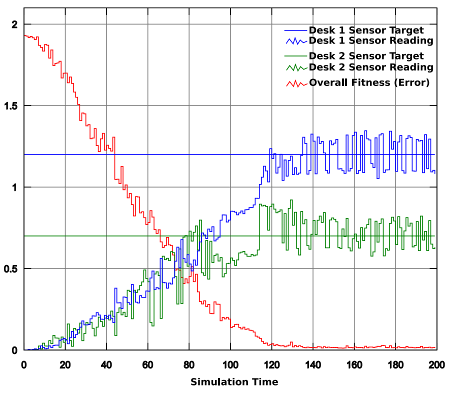


Fig. 2. Scenario 1 resembles the basic situation of powering up the lamps from darkness to a desired brightness

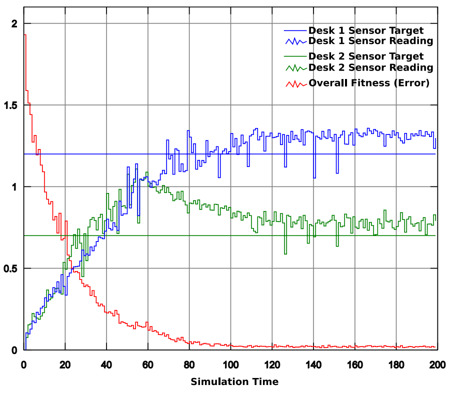


Fig. 3. Scenario 2 resembles Scenario 1, but with 100 instead of only 2 lamps

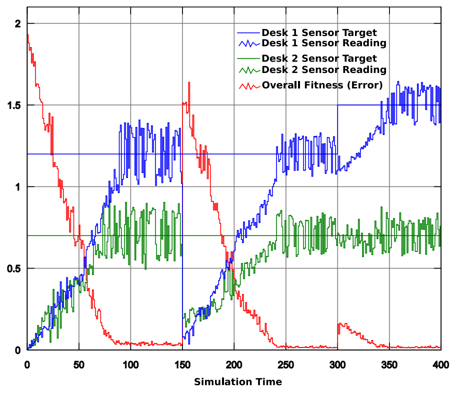


Fig. 4. Scenario 3 focuses on the ensemble behavior in more dynamic setups in which light sources might fail or join. At time $t = 150$ two of four lamps fail, at time $t = 300$ the second sensor target value is increased from $s_2^t = 1.2$ to $s_2^t = 1.5$.

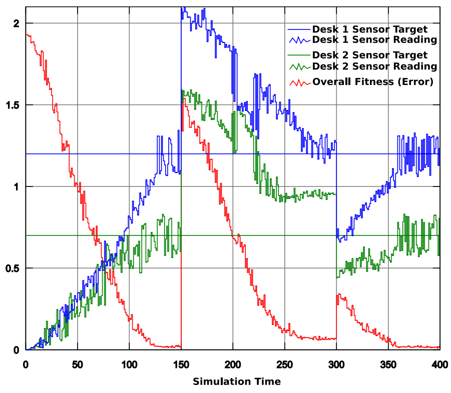


Fig. 5. This scenario models the influence of external modalities, which are outside the control of the ensemble. Especially, between $t = 150$ and $t = 300$, the external influence already exceeds the target value s_1^t and therefore prevents the system to reach the specified goal.

these runs, the figures always present the values from the run that was *worst* in the corresponding time step t.

All figures clearly indicate that the global system error decreases exponentially, and that thus after some adaption time, the AGE-P algorithm arrives at the specified target values, i.e., $s_i \approx s_i^t$. This is not only observable in the simple scenarios 1 and 2, but also in the more dynamic one (Fig. 4) in which the target values change over time. The only exception occurs in the fourth scenario (Fig. 5) between time steps 150 and 300. However, it has already been discussed above that AGE-P cannot reach the target values, since the external illumination is brighter than the specification demands. In other words, the small deviation from the optimum is not due to AGE-P but due to the physics; but even in this case, AGE-P returns to the optimum shortly after the reduction of the external illumination in time step $t = 300$.

It might be quite interesting to take a look at the scaling behavior of the proposed self-organization algorithm. Normally, an increasing number of components slows down the system convergence speed. However, a comparison of Fig. 2 and Fig. 3 indicates that a larger ensemble (i.e., 50 light sources per desk) reaches the optimal even faster than a smaller one (i.e, 2 light sources per desk). This effect is counter intuitive but probably due to a significantly increased number of actuator configurations that match the optimum sensor readings.

Rather than focusing on convergence speed, the application at hand focuses on a smooth adaption of its actuators. To this end, the step size σ should be set to rather small values; larger values would speed up the adaptation process, but would also induce significant fluctuations around the optimum, which might be rather annoying in real-world applications. Furthermore, a larger number of actuators, such as a total of 100 light sources used in scenario 2, requires a smaller step size, such as $\sigma = 0.02$. Otherwise, the fluctuations would be way too large to be acceptable. Therefore, future versions of AGE-P should employ a proper self-adaption scheme of the step size σ.

6 Conclusions

This paper has proposed a distributed evolutionary algorithm, called AGE-P, for the self-organization of smart-appliances ensembles. A key feature of this algorithm is that it does not maintain assembled genomes in the traditional sense. Rather, AGE-P physically distributes all gene values across all the devices, and evaluates only the resulting sensor modalities. Furthermore, the application of the mutation operators is done by the actuators rather than a central processing instance.

The presented simulation results indicate that the proposed method is suitable as the self-organization mechanism for smart-appliances ensembles. In addition to the required basic adaptation capabilities, the AGE-P framework scales well, and is also able to cope with the inherent system dynamics of those ensembles.

The experiments also show that the behavior of AGE-P depends on the chosen step size σ of the mutation operators, privately employed in every actuator. Therefore, future research will be devoted to the development of an adequate self-adaption mechanism.

Ongoing research is developing hardware equipment that consists of remotely controllable light sources as well as remotely readable sensors. All these devices are equipped with a wireless communication module. Preliminary results indicate that in the real-world, AGE-P also has to cope with varying time constants. For example, slight brightness changes may be faster than drastic brightness changes. Furthermore, these timing constants also depend on the chosen light source types and potentially other system parameters.

Acknowledgements

The authors gratefully thank Ralf Joost and Ulf Ochsenfahrt for encouraging discussions and valuable comments on draft versions of this paper. This work was supported in part by the DFG graduate school 1424.

References

1. Aarts, E.: Ambient intelligence: A multimedia perspective. IEEE Multimedia 11(1), 12–19 (2004)
2. Bäck, T., Hammel, U., Schwefel, H.-P.: Evolutionary Computation: Comments on the History and Current State. IEEE Transactions on Evolutionary Computation 1(1), 3–17 (1997)
3. Bry, F., Hattori, T., Hiramatsu, K., Okadome, T., Wieser, C., Yamada, T.: Context Modeling in OWL for Smart Building Services. In: Brass, S., Goldberg, C. (eds.) Tagungsband zum 17. GI-Workshop über Grundlagen von Datenbanken, Wörlitz, Germany, Gesellschaft für Informatik, pp. 38–42. Institute of Computer Science, Martin-Luther-Univerity Halle-Wittenberg (2005)

4. Chen, H., Perich, F., Finin, T., Joshi, A.: SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In: International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, MA (2004)
5. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Learning Intelligence. IEEE Press, NJ (1995)
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
7. Lee, C.-Y., Antonsson, E.K.: Variable Length Genomes for Evolutionary Algorithms. In: Whitley, L.D., Goldberg, D.E., Cantú-Paz, E., Spector, L., Parmee, I.C., Beyer, H.-G. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), p. 806 (2000)
8. Ramsey, C.L., De Jong, K.A., Grefenstette, J.J., Wu, A.S., Burke, D.S.: Genome Length as an Evolutionary Self-Adaptation. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) Proceedings of Fifth International Conference on Parallel Problem Solving from Nature (PPSN V), pp. 27–30 (1998)
9. Rechenberg, I.: Evolutionsstrategie (Frommann-Holzboog, Stuttgart) (1994)
10. Saha, D., Mukherjee, A.: Pervasive computing: A paradigm for the 21st century. IEEE Computer, 25–31
11. Schiffmann, W., Joost, M., Werner, R.: Application of Genetic Algorithms to the Construction of Topologies for Multilayer Perceptrons. In: Proceedings of Artificial Neural Networks and Genetic Algorithms, pp. 675–682 (1993)
12. Schwefel, H.-P.: Evolution and Optimum Seeking. John Wiley and Sons, NY (1995)
13. Weiser, M.: Some computer science issues in ubiquitous computing. Communications of the ACM 26(7), 75–84 (1993)

Adding Probabilistic Dependencies to the Search of Protein Side Chain Configurations Using EDAs

Roberto Santana[1], Pedro Larrañaga[2], and Jose A. Lozano[1]

[1] Intelligent Systems Group
[2] Department of Computer Science and Artificial Intelligence
University of the Basque Country
Paseo Manuel de Lardizábal 1, 20018. San Sebastian - Donostia, Spain
[3] Department of Artificial Intelligence, Technical University of Madrid,
28660 Boadilla del Monte, Madrid, Spain
roberto.santana@ehu.es, pedro.larranaga@fi.upm.es, ja.lozano@ehu.es

Abstract. The problem of finding an optimal positioning for the side chain residues of a protein is called the side chain placement or side chain prediction problem. It can be posed as an optimization problem in the discrete domain. In this paper we use an estimation of distribution algorithm to address this optimization problem. Using a set of 50 difficult protein instances, it is shown that the addition of dependencies between the variables in the probabilistic model can improve the quality of the solutions achieved for most of the instances considered. However, we also show that only when information about the known interactions between the residues is considered in the creation of the probabilistic model, the addition of the dependencies contributes to improve the quality of the solutions obtained.

Keywords: estimation of distribution algorithm, protein structure prediction, probabilistic models.

1 Introduction

Estimation of distribution algorithms (EDAs) [12,14,15] are evolutionary algorithms that use probability models instead of genetic operators. Probabilistic modeling allows EDAs to represent relevant features from the search space that can be automatically learned from the data using machine learning methods. The ability of EDAs to solve hard optimization problems and the capacity to extract previously unknown features of the problem domains have contributed to a recent upsurge of their applications in Bioinformatics [2,11,19,21].

In this paper, we address the important issue of the influence that the use of probabilistic dependencies has in the solution of the protein side chain placement problem. In [19], an EDA approach, based on the application of the univariate marginal distribution algorithm (UMDA) [15], has been applied to this protein problem. The UMDA approach uses the simplest probabilistic model where all variables are considered independent.

In EDAs, the capacity of the model to represent complex interactions between the problem components usually influences its accuracy. On the other hand, increasing the model complexity has also an associated computational cost. Therefore, central to the notion of efficient modeling is the achievement of an appropriate balance between the adequate complexity of the model (enough to represent the relevant features of the problem) and a feasible computational cost (the time and the storage requirements must be affordable).

Although UMDA's capacity of representation is highly limited, for a number of difficult protein side chain placement instances where other state-of-the-art algorithms failed to converge, UMDA was able to find better structures than the other algorithms [19].

In this paper, we investigate the question of whether the representation of interactions, by means of probability dependencies between the variables determines a real improvement in the efficiency of EDAs for the protein side chain placement problem. To this end, we apply a tree-based EDA and compare its results with UMDA. In addition, we introduce a proposal to improve the quality of the solutions found and diminish the computational burden of the algorithms by using the protein structure information available.

2 Protein Side Chain Placement Problem

We use X_i to represent a discrete random variable. A possible value of X_i is denoted x_i. Similarly, we use $\mathbf{X} = (X_1, \ldots, X_n)$ to represent an n-dimensional random variable and $\mathbf{x} = (x_1, \ldots, x_n)$ to represent one of its possible values.

Assuming that the position of the protein backbone is fixed, and considering fixed bond lengths, the location of the protein side chain residues can be completely determined by the sidechain dihedral angles. The problem of finding an optimal positioning for the side chain residues is called side chain placement or side chain prediction [13].

A way to address the problem is to constrain the search to the discrete space by means of discrete configurations of the angles, known as rotamers [6]. A rotamer, short for rotational isomer, is a single side chain conformation represented as a set of discrete values, one for each dihedral angle degree of freedom [6]. A rotamer library is a collection of rotamers for each residue type.

The inclusion of these discrete configurations implies an important problem reduction. The search for the protein structure is 'reduced' to the search of a set of rotamers (one for each residue) that optimizes the fitness function. However, the combinatorial problem is NP-hard [17] and, in general, the use of brute force algorithms is unaffordable.

In the protein side chain problem, variable X_i will represent the set of dihedral angles corresponding to the i-th residue. x_i will be interpreted as one of the indexed set of rotamer configurations from the rotamer library. Each configuration encodes one value for each of the dihedral angles of the i-th residue. The number of values of each variable will correspond to the number of rotamer configurations for the corresponding residue i (i.e. $x_i \in \{1, \ldots, k_i\}$, where k_i is the number of feasible rotamer configurations for residue i).

When the backbone is fixed, the energy of a sequence of n amino acids folded into a defined structure can be expressed as:

$$E(\mathbf{x}) = \sum_{i=1}^{n} E(x_i) + \sum_{i=1}^{n-1} \sum_{j>i}^{n} E(x_i, x_j), \tag{1}$$

where $E(x_i)$ represents the energy interaction between the rotamer and the backbone as well as the intrinsic self-energy of the rotamer. $E(x_i, x_j)$ is the interaction energy between all pairs of sidechains. For two sets of atoms, the interaction energy is the sum of the pairwise atom interactions. We have adopted the van der Waals energy function as implemented in [23]. This energy function approximates the repulsive portion of Lennard-Jones 12-6 potential. It penalizes steric clashes between atoms. The energy of residues that do not interact is zero for every possible rotamer configuration.

Different optimization approaches to optimal side chain prediction have been proposed. Among the most common approaches used are dead-end elimination (DEE) algorithms [5], the self consistent mean field approach (SCMF) [10], and side chain placement with rotamer library (SCWRL) [6]. Inference-based methods [23] can be also used to find the exact solutions of the side chain prediction problem but they may fail to converge for some difficult instances. In these cases, the application of heuristic approaches is necessary.

3 Estimation of Distribution Algorithms

We will work with positive probability distributions denoted by $p(\mathbf{x})$. Similarly, $p(\mathbf{x}_S)$ will denote the marginal probability distribution for $\mathbf{X}_S$, where $S \subset \{1, \ldots, n\}$.

In the EDA approach we follow, each individual represents one possible solution and it is encoded using the vector representation introduced above. The selection step is based on the evaluation of a predefined fitness function. A key characteristic and crucial step of EDAs is the construction of the probabilistic model. These models may differ in the order and number of the probabilistic dependencies that they represent.

UMDA is the simplest EDA approach to the problem treated in this paper. The probability assigned by the model to each solution is equal to the product of the variables' univariate probabilities. UMDA's estimation step consists of calculating the univariate frequencies of each value for every variable. In the sampling step, new solutions are generated by independently sampling each variable.

In this paper, we propose the application of a model that captures bivariate dependencies between the variables. This probabilistic model is based on a tree where each variable may depend on at most another variable that is called the parent. A probability distribution $p_{Tree}(\mathbf{x})$ that is conformal with a tree is defined as:

$$p_{Tree}(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | pa(x_i)) \tag{2}$$

where $Pa(X_i)$ is the parent of X_i in the tree, and $p(x_i|pa(x_i)) = p(x_i)$ when $Pa(X_i) = \emptyset$, i.e. X_i is the root of the tree. The distribution $p_{Tree}(\mathbf{x})$ itself will be called a tree model when no confusion is possible. Probabilistic trees are represented by acyclic connected graphs.

The construction of the tree structure from data implies the detection of the most important bivariate interactions between the variables. This can be done applying statistical independence tests [16] or methods based on the analysis of the mutual information between variables as in [1]. We follow the second approach using the tree-based EDA (Tree-EDA)[1].

Initially, the bivariate probabilities are calculated for every pair of variables. From these bivariate probabilities, the mutual information between variables is found. To construct the tree structure, an algorithm introduced in [4], that calculates the maximum weight spanning tree from the matrix of mutual information between pairs of variables, is used. Probabilistic logic sampling [8] is used to sample new solutions from the tree. New solutions are generated starting from the root of the tree and sampling each variable conditioned by its parent. More details about the algorithm and its computational cost could be found in [22]. Tree-EDA has been successfully applied to other classes of protein problems [18,21].

3.1 Using Problem Information to Increase Efficiency of EDAs

The energy function used to evaluate the protein side chain problem assigns a zero contribution to pairs of residues that do not interact. We could expect that the probabilistic dependencies between the corresponding pairs of variables will be weaker. Therefore, one variant of the tree learning algorithm constrains the calculation of bivariate probabilities and mutual information to those pairs of variables corresponding to residues that are interacting in the backbone (those that actually interact in the crystal structure). We call this algorithm Tree-EDA^r. During the learning phase of Tree-EDA^r, the computation of the mutual information is done only for the previously fixed subset of variables pairs. The tree structure only includes pairs of variables that belong to this subset.

This approach, which has been previously tested for other class of protein problems [18], helps to reduce the number of spurious correlations that contribute to deteriorate the accuracy of the models and negatively influence the efficiency of the search.

The computational complexity of EDAs is mainly dependent on the complexity of the learning algorithm, but also depends on the population size and number of generations needed for convergence, which are problem-dependent. While the computational complexity of UMDA is linear in the number of variables, for Tree-EDA it is quadratic [22]. Nevertheless, the use of problem structure, as done by Tree-EDA^r, reduces the time spent to learn the probabilistic model.

[1] C++ (EDA program) and Matlab (MATEDA) implementations of UMDA, Tree-EDA, and other EDAs are respectively available from endika@si.ehu.es and http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/

4 Experiments

First, we introduce the protein benchmark and the parameters used by the algorithms. Then, we explain how the experiments were designed. Finally, the results of the experiments are presented.

4.1 Protein Benchmarks

To test our algorithms, we started with a protein dataset of 493 X-ray crystal structures[2] with a resolution better than or equal to $2\AA$, an R factor below 20%, and a mutual sequence identity lower than 50%. Each protein consisted of 1-4 chains and up to 1000 residues. As a pre-processing step, we determined the instances in which the Goldstein criterion [5] eliminated all configurations but one, and those instances in which the inference-based algorithm for structure prediction (SPRINT) [23] converged.

SPRINT is one of the state-of-the-art algorithms for protein side chain placement. In [23], the energies obtained by SCWRL [3,6] (version 2.9) were reported to be strictly higher than those found by SPRINT in the small class of instances. Unfortunately, the SCWRL (version 3.0) implementation does not provide the energy values corresponding to solutions calculated by the algorithm. Proteins that were solved using the Goldstein criterion and those for which SPRINT converged were removed from the original database. The number of the remaining instances, which were used for our experiments, was 50. They serve as an appropriate testbed to focus the investigation of EDAs on a representative set of difficult instances were other efficient algorithms have failed.

4.2 Parameters of the Algorithms

To work, EDAs require the definition of several parameters. We have used the same settings for all instances of the problems treated. The quality of the results achieved by the algorithms will depend on these settings. Since, in this paper, we focus on the role played by dependencies, no attempt has been made to tune the parameters to achieve an optimal performance. The parameters of the EDAs have been set as follows. The population size was set at 5000. The maximum number of generations is 500. Truncation selection with parameter $T = 0.15$ has been used. In this selection scheme, the best $T \cdot N$ individuals of the population are selected to construct the probabilistic model. By setting a rather low value of truncation, a strong selection pressure is induced, forcing the algorithm to discard poor solutions as a faster pace.

We apply a replacement strategy called best elitism in which the selected population at generation t is incorporated into the population of generation $t + 1$, keeping the best individuals found so far and avoiding to revaluate their fitness function. The algorithm stops when the maximum number of generations

[2] These instances have been obtained from Chen Yanover's page: http://www.cs.huji.ac.il/∼cheny/proteinsMRF.html

is reached or the selected population has become too homogeneous (no more than 10 different individuals).

EDAs incorporate an additional problem reduction step to decrease the number of variables and their number of values. This step starts from the application of a dead-end elimination step [5], based on the iterative use of the Goldstein elimination criterion, which establishes a sufficient condition for rotamer configuration x_i to be absent from the optimal solution. When no condition that further eliminates rotamers can be established, the algorithm stops. This step considerably contributes to reduce the dimension of the search space, but the search space remains huge.

4.3 Design of the Experiments

To compare the results of the algorithms we conducted, in most of cases, 50 experiments for each instance and algorithm. For a number of complex instances, the number of experiments for each algorithm was reduced to 30. The performance of the algorithms was evaluated considering the fitness of the best solution found in each experiment, the best fitness among all the best solutions found, and the number of experiments in which the best fitness was found.

To determine whether differences between the fitness of the solutions found by the algorithms are statistically significant the Kruskal-Wallis test [9] was employed. The test significance level was 0.05. To compare the algorithms according to the best fitness, we determined for each instance, which was the best solution found by each algorithm in all the experiments done.

4.4 Numerical Results

We compared the quality of the solutions obtained by UMDA, Tree-EDA and Tree-EDA^r using the protein benchmark. Tables 1 and 2 shows the 27 instances for which Tree-EDA^r found solutions with energies strictly lower than those found by SPRINT, UMDA and Tree-EDA[3]. Table 1 shows the results corresponding to those instances for which 50 experiments were conducted and Table 2 those for which 30 experiments were done. The best solution found by Tree-EDA^r has higher energy than the best solution found by SPRINT in only 5 of the 50 instances. The table shows the size of each instance after the application of the Goldstein criterion (size), the number of experiments conducted (exp.), the best value of the fitness found in all the experiments (best), the number of times the best solution was found (S), and the average fitness (mean) of the best solutions of each experiment. The standard deviation of the EDAs results changed according to the instance used. For sake of space these values are not reported.

The application of the Kruskal-Wallis test found significant statistical differences between UMDA and Tree-EDA for 26 of the 50 instances. For 15 of these instances, UMDA was better than Tree-EDA. Significant statistical differences between UMDA and Tree-EDA^r were found for 45 of the 50 instances. For 43

[3] Detailed results for all the instances are available as additional documentation from http://www.sc.ehu.es/ccwbayes/EDA/EDAProteinProblems.html

Table 1. Results achieved by UMDA, Tree-EDA and Tree-EDA^r for the selected instances

			UMDA			Tree-EDA			Tree-EDA^r		
pdb id	*size*	*exp.*	*best*	*S*	*mean*	*best*	*S*	*mean*	*best*	*S*	*mean*
pdb1crz	75	50	626.41	1	627.25	626.12	7	627.54	626.12	24	626.56
pdb1ddt	146		754.93	1	760.02	754.30	1	760.88	753.38	26	754.05
pdb1dpe	185		727.37	2	750.51	725.83	1	739.73	725.50	18	729.94
pdb1gsk	208		939.94	1	947.77	934.79	1	945.74	934.01	12	935.62
pdb1h3n	318		1626.09	1	1639.00	1617.70	1	1653.47	1611.76	1	1625.08
pdb1jy1	144		861.92	1	870.34	856.88	4	860.76	856.84	33	856.92
pdb1kmo	241		925.90	1	943.09	890.85	1	924.32	875.20	1	886.86
pdb1kwh	207		972.11	1	988.21	960.73	2	980.09	959.17	1	965.53
pdb1nqe	189		570.29	1	593.49	563.36	1	588.67	563.30	16	566.70

of these instances, Tree-EDA^r outperformed UMDA. The statistical tests confirmed what can be seen from Tables 1 and 2: Tree-EDA^r consistently found better solutions than UMDA. However, the performance of Tree-EDA is not always better than UMDA. Furthermore, a detailed analysis of the experiments (see supplementary material), can reveal that UMDA beats the other two EDAs for the largest problem instances. This might be explained by the fact that the population sizes used by Tree-EDA and Tree-EDA^r are still insufficient to learn an accurate model of the interactions in very large problems. An exhaustive analysis of this question is beyond the scope and space limitations of this paper.

Table 2. Results achieved by UMDA, Tree-EDA and Tree-EDA^r for the selected instances

			UMDA			Tree-EDA			Tree-EDA^r		
pdb id	*size*	*exp.*	*best*	*S*	*mean*	*best*	*S*	*mean*	*best*	*S*	*mean*
pdb1dxr	353	30	1703.73	1	1722.79	1701.61	1	1716.89	1695.16	5	1699.57
pdb1dz4	288		875.77	1	884.79	868.92	1	880.96	867.01	3	872.16
pdb1d2e	281		1839.67	1	1847.65	1829.95	1	1841.83	1823.87	2	1829.18
pdb1e61	454		1936.92	1	1958.89	1942.94	1	1992.96	1911.46	1	1918.09
pdb1e6p	365		1681.67	1	1694.86	1681.93	1	1703.08	1673.47	1	1680.59
pdb1f60	123		537.42	3	540.78	536.69	1	540.68	535.14	6	536.41
pdb1fmj	294		1100.51	1	1121.42	1089.81	1	1105.23	1088.80	8	1092.90
pdb1fn9	239		989.51	3	993.92	988.82	1	1004.05	987.13	1	990.61
pdb1fnn	240		735.75	1	749.53	732.90	1	751.41	732.01	4	735.82
pdb1giq	265		806.53	1	823.58	801.38	1	815.62	800.07	3	800.95
pdb1h3f	206		785.56	1	795.11	784.95	1	794.09	782.98	7	785.68
pdb1h4r	227		825.64	1	830.12	816.96	1	824.09	815.84	8	817.45
pdb1h80	229		1036.90	1	1040.45	1034.96	1	1039.07	1034.77	9	1035.15
pdb1iqc	288		1538.37	1	1546.85	1531.80	1	1536.20	1530.18	5	1532.12
pdb1jmx	285		1518.10	1	1545.51	1510.21	1	1534.77	1504.70	3	1510.91
pdb1j3b	289		1600.16	1	1625.67	1592.75	1	1616.81	1584.68	2	1598.96
pdb1j8f	329		957.08	1	964.50	942.69	3	954.67	942.62	16	943.56
pdb1lqt	268		935.95	1	967.32	926.16	1	941.13	926.16	12	927.22
pdb1lsh	350		1125.04	1	1135.00	1120.77	1	1132.27	1117.76	1	1119.78
pdb1tki	164		858.67	1	867.24	856.62	1	860.97	855.56	5	856.98

Additional experiments were conducted to investigate the structural differences between the best solutions found by UMDA and those obtained using Tree-EDA and Tree-EDA^r. This type of experiments is illustrated using dimer protein pdb1tki. This protein has two symmetric chains and the number of residues before the application of the Goldstein criterion is 576. The energies corresponding to best side chain structures found by UMDA, Tree-EDA and Tree-EDA^r were respectively 858.67, 856.62 and 855.56.

The structures found by Tree-EDA and Tree-EDA^r are respectively shown in Figure 1 (left) and Figure 1 (right). The protein structures found by UMDA and Tree-EDA are different in only 7 residues. These residues are identified by its corresponding sequence number in Figure 1 (left). Their rotamer configurations are accountable for the improvement in the energy. In the figure, those residues that interact are linked. On the other hand, the structures found by UMDA and Tree-EDA^r are different in only 9 of the residues which are identified in Figure 1 (right). Also in this case, every residue is linked at least to another one.

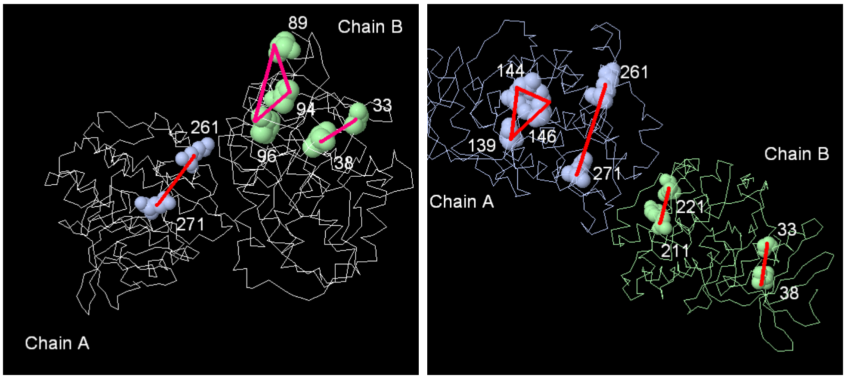


Fig. 1. Backbone and best side chains configurations found by EDAs for a subset of protein pdb1tki's residues. Learned by Tree-EDA (left). Learned by Tree-EDA^r (right).

In general, changes between the rotamer configurations of the best solutions found by UMDA and those found using probabilistic dependencies occur for clusters of interacting residues. UMDA is supposed to find the optimal configurations for pair of residues with weak interactions but it is not able to find the optimal configurations for clusters of variables that strongly interact. For several of these cases, the use of probabilistic dependencies improve the results. In this problem, the contrastive analysis of the best solutions found with and without the use of dependencies helps to identify clusters of variables with strong patterns of interactions which in turn correspond to residues that interact in the protein structure.

5 Conclusions and Future Work

The results presented in this paper show how results achieved in the protein side chain placement problem can be noticeably improved by the use of probabilistic models able to represent interactions between the variables. The results obtained by Tree-EDA^r could be further improved by the application of local search methods as those used in [20] to refine the solutions found by UMDA. Other EDAs could be applied, however the application of EDAs that use more complex probabilistic models, such as Bayesian networks or Markov networks, does not seem to be a good option due to the high cardinality of the problem variables.

Finally, we point to the fact that contrastive analysis of solutions found using different classes of models could be added to the set of available techniques [7,18,21] used to extract problem information from the probabilistic models learned during the search.

Acknowledgements

The authors thank Chen Yanover for providing the set of instances used in the experiments. This work has been partially supported by the Etortek, Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2005-03824 and Consolider Ingenio 2010 - CSD2007-00018 projects (Spanish Ministry of Education and Science) and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

1. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: Proceedings of the 14th International Conference on Machine Learning, pp. 30–38. Morgan Kaufmann, San Francisco (1997)
2. Belda, I., Madurga, S., Llorá, X., Martinell, M., Tarragó, T., Piqueras, M., Nicolás, E., Giralt, E.: ENPDA: An evolutionary structure-based de novo peptide design algorithm. Journal of Computer-Aided Molecular Design 19(8), 585–601 (2005)
3. Canutescu, A.A., Shelenkov, A.A., Dunbrack, R.L.: A graph-theory algorithm for rapid protein side-chain prediction. Protein Science 12, 2001–2014 (2003)
4. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3), 462–467 (1968)
5. De Maeyer, M., Desmet, J., Lasters, I.: The dead-end elimination theorem: Mathematical aspects, implementation, optimization, evaluation, and performance. Methods in Molecular Biology 143, 265–304 (2000)
6. Dunbrack, R.L.: Rotamer libraries in the 21st century. Current Opinion in Structural Biology 12, 431–440 (2002)
7. Echegoyen, C., Lozano, J.A., Santana, R., Larrañaga, P.: Exact Bayesian network learning in estimation of distribution algorithms. In: Proceedings of the 2007 Congress on Evolutionary Computation CEC 2007, pp. 1051–1058. IEEE Press, Los Alamitos (2007)
8. Henrion, M.: Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In: Lemmer, J.F., Kanal, L.N. (eds.) Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence, pp. 149–164. Elsevier, Amsterdam (1988)
9. Hsu, J.C.: Multiple Comparisons: Theory and Methods. Chapman and Hall, Boca Raton (1996)
10. Koehl, P., Delarue, M.: Building protein lattice models using self consistent mean field theory. Journal of Chemical Physics 108, 9540–9549 (1998)

11. Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J.A., Armañanzas, R., Santafé, G., Pérez, A., Robles, V.: Machine learning in bioinformatics. Briefings in Bioinformatics 7, 86–112 (2006)
12. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Boston (2002)
13. Lee, C., Subbiah, S.: Prediction of protein side-chain conformation by packing optimization. Journal of Molecular Biology 217, 373–388 (1991)
14. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.): Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer, Heidelberg (2006)
15. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
16. Pelikan, M., Mühlenbein, H.: The bivariate marginal distribution algorithm. In: Roy, R., Furuhashi, T., Chawdhry, P. (eds.) Advances in Soft Computing - Engineering Design and Manufacturing, London, pp. 521–535. Springer, Heidelberg (1999)
17. Pierce, N.A., Winfree, E.: Protein design is NP-hard. Protein Engineering 15(10), 779–782 (2002)
18. Santana, R., Larrañaga, P., Lozano, J.A.: The role of a priori information in the minimization of contact potentials by means of estimation of distribution algorithms. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 247–257. Springer, Heidelberg (2007)
19. Santana, R., Larrañaga, P., Lozano, J.A.: Side chain placement using estimation of distribution algorithms. Artificial Intelligence in Medicine 39(1), 49–63 (2007)
20. Santana, R., Larrañaga, P., Lozano, J.A.: Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem. Journal of Heuristics (to appear, 2008)
21. Santana, R., Larrañaga, P., Lozano, J.A.: Protein folding in simplified models with estimation of distribution algorithms. IEEE Transactions on Evolutionary Computation (to appear, 2008)
22. Santana, R., Ochoa, A., Soto, M.R.: The mixture of trees factorized distribution algorithm. In: Spector, L., Goodman, E., Wu, A., Langdon, W., Voigt, H., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., Burke, E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2001, pp. 543–550. Morgan Kaufmann Publishers, San Francisco (2001)
23. Yanover, C., Weiss, Y.: Approximate inference and protein-folding. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems 15, pp. 1457–1464. MIT Press, Cambridge (2003)

A Scalable Formal Framework for Analyzing the Behavior of Nature-Inspired Routing Protocols

Muhammad Shahzad, Saira Zahid, and Muddassar Farooq

Next Generation Intelligent Networks Research Center (nexGIN RC),
National University of Computer and Emerging Sciences (NUCES),
Islamabad 44000, Pakistan
{mshahzad86,sairazahid12}@gmail.com, muddassar.farooq@nu.edu.pk

Abstract. Nature-inspired routing algorithms for fixed networks is an active area of research. In these algorithms, *ant-* or *bee-agents* are deployed for collecting the state of a network and providing them to *autonomous* and *fully distributed* controllers at each network node. In these routing systems the agents, through local interactions, self-organize to produce system-level behaviors which show adaptivity to changes and perturbations in the network environment. The formal modeling of such fully *self-organizing*, *distributed* and *adaptive* routing systems is a difficult task. In this paper, we propose a scalable formal framework that has following desirable features: (1) it models important performance metrics: throughput, delay and goodness of links, (2) it is scalable to any size of topology, (3) it is robust to changing network traffic conditions. The proposed framework is utilized to model a well-known *BeeHive* protocol which is further validated on NTTNeT (a 57 node topology). To the best of our knowledge, this is the first formal framework that has been validated on such a large topology.

1 Introduction

Nature-inspired routing protocols is an active area of research and has lead to a number of state-of-the-art algorithms like AntNet [2] and BeeHive [10]. These *bottom-up* algorithms are characterized by the presence of a set of distributed, autonomous, minimalist agents. These agents, through local interactions, self-organize to produce system-level behaviors which show adaptivity to changes and perturbations in the external environment. Moreover, these algorithms are usually resilient to minor internal failures and losses of agents, and scale quite well by virtue of their modular and fully distributed design [4]. Farooq and Di Caro have highlighted a serious shortcoming of these algorithms which is the "difficulty, somehow intrinsic to fully distributed and stochastic bottom-up approaches, to provide formal guarantees in terms of dependability" [4]. One of the most important reason which contributes towards this difficulty is the lack of formal models of these algorithms, which eventually leads to studying their behavior in simulators only.

To the best of our knowledge, no comprehensive research, except the preliminary work reported in [1], has been conducted in developing a formal framework that provides insight into the behavior of Nature-inspired routing protocols for fixed networks. In our earlier work [11], we proposed an elementary framework that utilizes stochastic recursive functions to model the behavior of *BeeHive*. We validated our model on relatively small topologies. The major contributions of this paper over our previous model are: (1) incorporating the hybrid clustering model of *BeeHive* that views a given network topology as overlapping *foraging zones* and non overlapping *foraging regions*, (2) the ability to incorporate any number of traffic sources and sinks (3) the ability to model varying traffic conditions ranging from low traffic loads to congested traffic loads, and (4) making the framework to scale to any size topology. We believe that these enhancements will help the designers to systematically study the scalability of a routing protocol in significantly short design cycles. It is to be noted that Farooq has shown in [3] that running a single experiment, even on high power machines, on a network topology of 1050 nodes takes $4-5$ days. While in our framework, it takes less than 9 minutes. Moreover, an important conclusion of our validation process is: *The behavior of* BeeHive *and relevant performance parameters estimated by our formal framework match with the behavior and results obtained from network simulations respectively.*

Organization of Paper. The rest of the paper is organized as follows. Section 2 provides a brief overview of BeeHive to make the paper self contained. We introduce our analytical framework in Section 3. We then discuss the outcome of our validation process on NTTNeT topology. Finally we conclude the paper with an outlook to our future research.

2 Overview of *BeeHive*

[1]*BeeHive* has been proposed by Wedde et al. [3], [10]. The algorithm has been inspired by the communication language of honey bees. Each node periodically sends a *bee agent* by broadcasting the replicas of it to each neighbor site. The replicas explore the network using priority queues and they use an estimation model to estimate the propagation and queuing delay from a node, where they are received, to their launching node. Once the replicas of the same agent arrive at a node via different neighbor sites of the node, they exchange routing information to model the network state at this node. Through this exchange of information by the replicas at a node, the node is able to maintain a quality metric for reaching destinations via its neighbor sites. The algorithm utilizes just forward moving agents and, as opposed to *AntNet*, no statistical parameters are stored in the routing tables. In *BeeHive*, a network is divided into *Foraging Regions* and *Foraging Zones*. Each node belongs to only one *Foraging Region*. Each *Foraging Region* has a representative node. A *Foraging Zone* of a node consists of all the nodes from whom a replica of an agent could reach this node in predefined number of hops. This predefined number of hops is termed as *short distance bee hop limit*. This approach significantly reduces the size of the routing table as compared to *AntNet* because each node maintains detailed routing information only about reaching the nodes within its *Foraging Zone* and for reaching the representative nodes of the *Foraging Regions*. In this way, a data packet, whose destination is beyond the *Foraging Zone* of a node, is forwarded in the direction of the representative node of the *Foraging Region* containing the destination node. The next hop for a data packet at a node is selected in a probabilistic

[1] This summary has been reproduced from our earlier paper [9].

Table 1. Symbols used in the paper

g_{ind}	goodness of a link (i, n) to reach d	ld_{in}	total delay for link (i, n)
pd_{in}	propagation delay for link (i, n)	tx_{in}	transmission delay for link (i, n)
q_{in}	queuing delay for the link (i, n)	x	iteration index
ξ_{id}	rate of the data packets entered by i to reach d	η_{id}	flow of data through i to reach d
$\mathfrak{lf}_{ind}$	flow of data on link (i, n) to reach d	L_{in}	total flow of traffic on link (i, n)
S_{in}	service rate of the link (i, n)	cd_{nd}	cumulative delay from node n to node d
td_{ind}	total delay from node i to d through n	$\mathcal{T}_{cum}$	cumulative throughput of the network
$\mathcal{V}$	set of all nodes in network	$\mathcal{N}_i$	set of all the neighbours of node i
$\mathcal{N}_{id}^{e}$	set of all the effective neighbours i to reach d	Θ_i	neighbor set operator
h	Short Distance Bee hop limit	h_{min}	minimum hops to reach destination
$\mathcal{Z}_i$	set of all the nodes in foraging zone of node i	R	set of all representative nodes
$\mathcal{Y}_{id}$	set of links constituting optimal path	q_{avg}	avergae queuing delay

fashion depending upon the goodness of each neighbor for reaching the destination. *BeeHive* is also fault-tolerant to crashing of routers. The interested reader will find more details in [10][3].

3 Analytical Model

We now discuss the most important aspect of our formal model that deals with the analytical modeling of *foraging zones* and *foraging regions*. This will help us in analytically doing the scalability analysis of *BeeHive* algorithm. We believe that this framework will obviate the need to conduct lengthy very time-consuming simulations, which may last even months [3], in order to study the behavior of an algorithm on large topologies. An interested reader may refer to [3][10] to study the motivation behind the concepts of *foraging zones* and *foraging regions*. We now define *neighbor set operator* that will help in modeling these concepts.

Definition 1 (Neighbor set operator). Neighbor set operator, $\Theta_i(x)$, is defined as:

$$\Theta_i(x) = \mathcal{N}_{\left(\Theta_i(x-1)\right)} \tag{1}$$

$x = 0 \Rightarrow \Theta_i(0) = \{i\}$, represents the set containing the current node i
$x = 1 \Rightarrow \Theta_i(1) = \mathcal{N}_i$, represents the set of neighbors of node i
$x = 2 \Rightarrow \Theta_i(2) = \mathcal{N}_{(\mathcal{N}_i)}$, represents the set of neighbors of the neighbors of node i
$\vdots \quad \vdots \quad \vdots \quad \vdots$

Foraging Zones. We now present our model for generating *foraging zones* of every node in the network. Now consider a node i (where i is also its address) in the network that launches *short distance bee agents* that can traverse h hops in the network starting from node i. Without loss of generality, we assume that the smallest address of any node in a given network is 1 and the largest address is $|\mathcal{V}|$, where the size of set $\mathcal{V}$ is total number of nodes in the network. We now define a family of sets $\mathcal{E}_i(x)$, which represents all links that can be possibly traversed by *short distance bee agents* launched by node i. An ordered pair (a, b) always represents a unidirectional link from node a to node b in rest of the paper.

$\forall a, b \in \mathcal{V}$

$$\mathcal{E}_i(x) = \{(a,b) | a \in \Theta_i(x) \wedge b \in \Theta_i(x+1)\} \qquad x = 0, 1, 2 \ldots h-1 \tag{2}$$

The elements of this family of sets are combined into one single set $\mathcal{A}_i$ as:

$$\mathcal{A}_i = \cup \mathcal{E}_i(x) \qquad x = 0, 1, 2 \ldots h-1 \tag{3}$$

The elements in the set $\mathcal{A}_i$ are in the form of ordered pairs that represent all the links traversed by *short distance bee agents* launched by node i. Now we define a set $\mathcal{Z}_i$ that contains all the nodes that are within h hops of the node i, and this set, therefore, represents *foraging zone* of node i as per its definition in [3][10].
$\forall (a, b) \in \mathcal{A}_i$

$$\mathcal{Z}_i = \{z | z = a \vee z = b\} \tag{4}$$

Foraging Regions. Now we model *foraging regions* of a network. We assume that propagation and transmission delays of all links in the network are the same. This assumption simplifies our formal model but in our future work we want to model a heterogenous network of varying delays and bandwidths to make it more realistic. would render our model We define a set T that initially contains all nodes in the networks and hence is equal to $\mathcal{V}$. Now we define set R (just an empty set at startup) that will contain all the *representative nodes* in a given network once the network is successfully portioned into *foraging regions*. We represent the *foraging region* of node i by a set $\mathcal{FR}_i$. Recall that the smallest node address is 1, therefore, $\mathcal{FR}_0 = \phi$. The iterative process of *foraging region* formation is given in the Algorithm 1:

Algorithm 1. make foraging regions

```
  while $T(x) \neq \phi$ do
    $\mathcal{FR}_i = \{z | z \in \mathcal{Z}_i\} - \sum_{n=0}^{i-1} \mathcal{FR}_n$        $i$ is the smallest value in $T$
    $R(x) = R(x-1) \cup \{i\}$
    $T(x) = T(x-1) - \mathcal{FR}_i$
  end while
```

The *foraging region* formation process is stopped once T becomes an empty set. Consequently, the set R will contain all the *representative nodes*. Note that $\forall i \in R, \mathcal{FR}_i$ is the *foraging region* of node i and $\forall i \notin R, \mathcal{FR}_i = \phi$.

Network Traffic Model. Now we present the formal framework for the network traffic in *BeeHive* protocol [10]. The network traffic is modeled using three important parameters: (1) g_{ind}, goodness of a link from node i to node n to reach a destination d, (2) td_{ind}, total delay experienced by a packet (propagation+queuing+transmission) to reach the destination node d from the source node i through its neighbor n, and (3) $\mathcal{T}_{cum}$, cumulative throughput of the network . We have formally modeled these parameters for very small topologies in our earlier basic framework [11]. In our current work, we now scale our model to large topologies.

Goodness. Recall that g_{ind} is the probability that a packet will be switched to neighbor n if its destination is d. This stochastic packet switching algorithm is an important component of all Nature-inspired algorithms [3][8]. We now define the term *effective neighbors* of a node i.

Definition 2 (Set of effective neighbors). The set of effective neighbors of node i ($\mathcal{N}^e_{id}$) to reach destination d is defined as:

$$\mathcal{N}^e_{id} = \{n | n \in \mathcal{N}_i \wedge n \in \mathcal{Z}_d\} \tag{5}$$

This definition ensures that if the current node is within the *foraging zone* of the destination then only those neighbors are selected as the next hop that are in the *foraging zone* of the destination. Now we model g_{ind}:

$$g_{ind}(x) = \begin{cases} \frac{\frac{1}{td_{ind}(x-1)}}{\sum_{k \in \mathcal{N}^e_{id}} (\frac{1}{td_{ikd}(x-1)})} & d{\in}\mathcal{Z}_i{\wedge}n{\in}\mathcal{N}^e_{id}{\wedge}i{\neq}n{\wedge}i{\neq}d \\ \frac{\frac{1}{td_{ind}(x-1)}}{\sum_{k \in \mathcal{N}_i} (\frac{1}{td_{ikd}(x-1)})} & d{\in}(R{-}\mathcal{Z}_i){\wedge}n{\in}\mathcal{N}_i{\wedge}i{\neq}n{\wedge}i{\neq}d \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

This definition successfully caters for three cases: (1) when the current node is within the *foraging zone* of the destination, (2) when the current node is not within the *foraging zone* of the destination, and therefore, in this case a packet is forwarded towards the *representative node* of the *foraging region* containing the destination node, and (3) unreachable destination. Now we model the node traffic.

Node traffic. We define the amount of traffic passing through a node as its node traffic. A node is allowed to generate data packets that can be destined for any other node in the network. Let ξ_{id} be the traffic generated by node i for destination d. We add to ξ_{id} the traffic that is transiting the current node i for destination d in order to get the total traffic, $\eta_{id}(x)$, for the destination:

$$\eta_{id}(x) = \begin{cases} \xi_{id} + \sum_{k \in \mathcal{N}^e_{id}} \eta_{ki}(x) g_{kid}(x) & i{\in}\mathcal{V}, d{\in}\mathcal{Z}_i{\wedge}i{\neq}d{\wedge}k{\neq}d \\ \xi_{id} + \sum_{k \in \mathcal{N}_i} \eta_{ki}(x) g_{kid}(x) & i{\in}\mathcal{V}, d{\in}(R{-}\mathcal{Z}_i){\wedge}i{\neq}d{\wedge}k{\neq}d \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In this way, a node traffic is modeled as probabilistic recursive functions in which the rates of packet arrival and departure are dependent on the goodness of incoming and outgoing links respectively. Finally, the current node will act as a sink node for the packets destined for it.

Link flow. The amount of traffic flowing on a link is defined as its link flow. It can be modeled as a function of the node traffic and the goodness of its neighbors. We now define $\mathfrak{lf}_{ind}$ which represents the fraction of the link flow of (i, n) for destination d as:

$$\mathfrak{lf}_{ind}(x) = \begin{cases} \eta_{id}(x) g_{ind}(x) & i{\in}\mathcal{V}, (n{\in}\mathcal{N}^e_{id} \text{ iff } d{\in}\mathcal{Z}_i) \vee (n{\in}\mathcal{N}_i \text{ iff } d{\in}(R{-}\mathcal{Z}_i)) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The overall link flow L_{in} of (i, n) is simply a sum of all its fractions $\mathfrak{lf}_{ind}$:

$$L_{in}(x) = \sum_{\forall d \in \mathcal{V}} \mathfrak{lf}_{ind}(x) \tag{9}$$

Calculation of delays. In computer networks, the arrival of packets is assumed to follow Poisson distribution and the queues that are built as a consequence are M/M/1 queues [5], [6], [7]. Moreover the service rates can be pre-assigned because they are independent of the network conditions. We can calculate the queuing delay associated with every traversed link by utilizing the M/M/1 queuing theory if we know the arrival and departure rates. Representing the service rate of (i, n) by S_{in}, we calculate the queuing delay on this link represented by q_{in} from the equation given below [6]:

$$q_{in}(x) = \begin{cases} \frac{1}{S_{in} - L_{in}(x)} & L_{in} < S_{in} \\ \infty & L_{in} \geq S_{in} \end{cases} \tag{10}$$

This represents only the queuing delay. The data packets experience transmission delay and propagation delay along with the queuing delay on any link. We represent the transmission delay from node i to node n by tx_{in} and propagation delay by pd_{in}. The total delay experienced by the packets on (i, n) is termed as the link delay and is represented by ld_{in}:

$$ld_{in}(x) = tx_{in} + q_{in}(x) + pd_{in} \tag{11}$$

The packet that is currently at node i will be switched to node n according to the goodness of that neighbor. Once the packet has reached this intermediate node n, it again might have multiple paths to reach the destination node through its neighbors. Therefore, we calculate the cumulative delay cd_{nd} from this node n to the destination d as:

$$cd_{nd}(x) = \begin{cases} \sum_{k \in \mathcal{N}^e_{nd}} \big(g_{nkd}(x)\big)\big(ld_{nk}(x) + cd_{kd}(x)\big) & d \in \mathcal{Z}_n \\ \sum_{k \in \mathcal{N}_n} \big(g_{nkd}(x)\big)\big(ld_{nk}(x) + cd_{kd}(x)\big) & d \in (R - \mathcal{Z}_n) \\ 0 & n = d \\ \infty & \text{otherwise} \end{cases} \tag{12}$$

Now we represent the total delay from node i to destination d as a sum of the link delay and cumulative delay.

$$td_{ind}(x) = ld_{in}(x) + cd_{nd}(x) \tag{13}$$

The goodness for the next iteration by using (6) is:

$$g_{ind}(x+1) = \begin{cases} \frac{\frac{1}{td_{ind}(x)}}{\sum_{k \in \mathcal{N}^e_{id}} \left(\frac{1}{td_{ikd}(x)}\right)} & d \in \mathcal{Z}_i \wedge n \in \mathcal{N}^e_{id} \wedge i \neq n \wedge i \neq d \\ \frac{\frac{1}{td_{ind}(x)}}{\sum_{k \in \mathcal{N}_i} \left(\frac{1}{td_{ikd}(x)}\right)} & d \in (R - \mathcal{Z}_i) \wedge n \in \mathcal{N}_i \wedge i \neq n \wedge i \neq d \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

So starting from the goodness value in (6) we reached (14). The iterations are stopped once the goodness values reach a steady state.

Throughput. The throughput is defined as the number of data packets successfully delivered to their destinations in unit interval of time. We can get the overall throughput $\mathcal{T}_{cum}$ by taking a sum of throughput of each node which is given by:

$$\mathcal{T}_{cum} = \sum_{\forall i \in \mathcal{V}} \sum_{\forall d \in \mathcal{V}} \left(\sum_{n \in \mathcal{N}_i} \frac{(g_{ind})(\xi_{id})}{td_{ind}} \right) \tag{15}$$

Optimal Path. Now we model the optimal path in terms of number of hops between any given source and destination pair. We consider the case when the destination is within the same *foraging zone* as of the source. We will show in Section 4 that most of the packets in *BeeHive* follow optimal path under low traffic loads and the load is distributed on multiple paths under high traffic loads. In order to formally model this optimal path, we first calculate the minimum hops h_{min} path between a given source and destination. For this purpose we find a set of all the possible links, $\mathcal{C}_{id}$, within the h_{min} hops from the source node, that lead to the destination. The minimum number of iterations in which the link set, $\mathcal{C}_{id}$ constitutes the destination node is basically the minimum hop number h_{min} and this is the optimal path $\mathcal{Y}_{id}$.

$$\mathcal{C}_{id} = \Big\{\mathcal{E}_i(x)|\Big((a,b) \in \mathcal{E}_i(h_{min})|b = d\Big)\Big\} \qquad {}_{x=0,1,2\ldots,h_{min}\wedge h_{min}<h} \tag{16}$$

$$\mathcal{Y}_{id} = \{(a_n, b_n)|(a_n, b_n) \in \mathcal{E}_i(n) \wedge a_n = b_{n-1} \wedge b_{h_{min}} = d\} \qquad {}_{n=0,1,2\ldots,h_{min}} \tag{17}$$

Pseudo code. At the end we present a pseudo code showing the sequence of steps to calculate the performance parameters discussed above.

Algorithm 2. Formal Model

```
generate foraging zones using (2) to (4)
generate foraging regions using Algorithm 1
while goodness values are not stabilized do
    calculate goodness using (6)
    calculate node traffic using (7)
    calculate link flows using (8)
    calculate total delay using (13)
    calculate throughput using (15)
end while
```

4 Empirical Verification of Formal Model

We will now validate our formal model on a well-known NTTNet shown in Figure 1. It is a non-balanced oblong network with a low degree of connectivity. We also simulated the reported scenarios in OMNeT++ simulator and then compared the estimated performance metrics of our formal framework with the metrics obtained from OMNeT++. We compared a number of parameters: (1) *foraging zones* for all the nodes, (2) *foraging regions* and *representative nodes* of the network, (3) goodness values of the links, (4) average total delay of the packets, (5) average queuing delay of the packets, (6) cumulative throughput of the network, (7) average hops of the data packets, and (8) optimal path. To the best of our knowledge, only our proposed formal framework is validated at such a large topology and for so many relevant performance metrics. The results of our validation framework clearly indicate that the performance metrics estimated by our formal framework are within an acceptable deviation of the values obtained from OMNeT++.

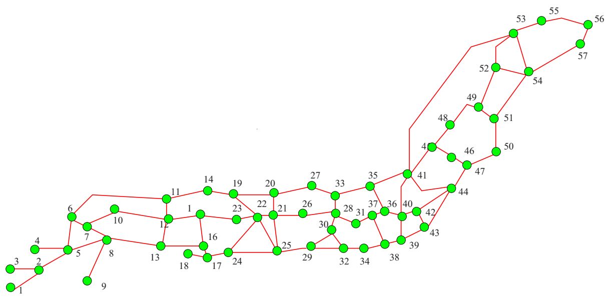


Fig. 1. NTTNeT

Foraging zones. We now compare the *foraging zones* formed by our formal model, by utilizing (2),(3) and (4), with the *foraging zones* of OMNeT++. Note that we used a *short distance bee hop limit* of 6. We tabulate the *foraging zones* of node 9, 12, 18 and 57 (just to show few examples) in Table 2. One can see that our formal model has accurately captured the process of zones formation.

Foraging regions. In Table 3 one can see that our formal model shows that with a *short distance bee hop limit* of 6 NTTNeT is partitioned into 5 regions with 1, 15, 27, 41, and 55 as the elected *representative nodes* of the regions. Again one can see the same results for OMNeT++.

Table 2. Foraging Zones

Node Number	Formal Model	OMNeT++
9	1 to 8, 10 to 19, 22 to 25	1 to 8, 10 to 19, 22 to 25
12	1 to 11, 13 to 30, 32, 33	1 to 11, 13 to 30, 32, 33
57	27,28,33,35 to 56	27,28,33,35 to 56

Table 3. Foraging Regions

Rep Node	Formal Model	OMNeT++
1	1 to 14,19	1 to 14,19
15	15 to 18,20 to 26	15 to 18,20 to 26
27	27 to 40	27 to 40
41	41 to 50,53	41 to 50,53
55	55,56	55,56

Goodness. In order to demonstrate the correctness of our goodness model, we now show the goodness values of the links for various source destination pairs. Just to cite an example, we plot the goodness values of different *effective neighbors* of node 28 to reach destination node 12 in Figure 2. The most important observation is that the goodness values reach a steady state after few iterations. Once the transient state finishes then the goodness values estimated by our formal model closely match with those obtained from OMNeT++ simulations.

We can see that the goodness value settles to zero for the link (28,31). It is obvious from Table 2, that node 31 is not in the foraging zone of node 12, therefore, it is not an effective neighbor (see Definition 2).

Cumulative throughput. We now report in Table 4 the cumulative throughput obtained from our formal model. We do that for four different values of mean packet inter-arrival time (MPIA): 0.005, 0.035. 0.25 and 1.0. We also tabulate the cumulative throughput obtained from OMNeT++ simulations for the same MPIA values. One can see in Table 4 that the throughput values, estimated by our formal model, are approximately the same as obtained from OMNeT++ simulations.

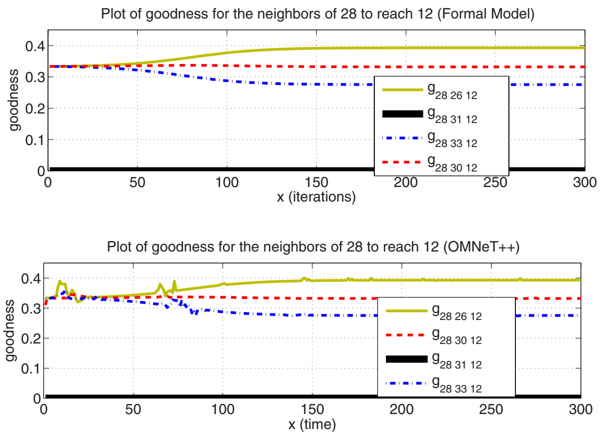


Fig. 2. A comparison of goodness between OMNeT++ and formal model for the neighbors of 28 to reach 12

Average total delay. We now show (see Table 4) that the average packet delays, td_{avg}, estimated by our formal, in NTTNeT for above-mentioned four MPIA values, are approximately the same as obtained from OMNeT++ simulations. The MPIA is used in this example as:

$$MPIA = \frac{1}{\xi_{id}} \tag{18}$$

Average queuing delay. Similarly, the average queuing delays, q_{avg}, obtained from our formal model, for different MPIA values also match with the average queuing delays of OMNeT++ simulations (see Table 4).

Average hops. One can see in Table 4 that our formal model correctly estimated the average number of hops, h_{avg} that a data packet took to reach its destination.

Table 4. A comparison of performance values of our Formal Model with OMNeT++ for NTTNeT

	Formal Model				OMNeT++			
MPIA	$\mathcal{T}_{cum}$	td_{avg}	q_{avg}	h_{avg}	$\mathcal{T}_{cum}$	td_{avg}	q_{avg}	h_{avg}
0.005	85.13	40.83	3.48	8.145	87.09	42.03	3.41	8.013
0.035	12.13	13.62	0.010	7.958	12.45	13.51	0.0093	7.893
0.25	1.64	12.46	0	7.42	1.75	12.5	0	7.37
1	0.41	11.9	0	7.127	0.44	12	0	7.01

Table 5. Percentage traffic on optimal path

i-*d* pair	Paths	Percentage Traffic			
		Formal Model		OMNeT++	
		0.005	1.0	0.005	1.0
21-28	$21 \rightarrow 26 \rightarrow 28$(optimal path)	35	93	35	84
	$21 \rightarrow 20 \rightarrow 27 \rightarrow 33 \rightarrow 28$	33	3	31	7
	$21 \rightarrow 25 \rightarrow 29 \rightarrow 30 \rightarrow 28$	30	2	32	8
32-38	$32 \rightarrow 34 \rightarrow 38$(optimal path)	41	96	40	89
	$32 \rightarrow 30 \rightarrow 28 \rightarrow 31 \rightarrow 37 \rightarrow 38$	35	2	36	7
	$32 \rightarrow 30 \rightarrow 28 \rightarrow 33 \rightarrow 35 \rightarrow 36 \rightarrow 37 \rightarrow 38$	21	1	22	3

Optimal path. We now show (see Table 5) that a large percentage of packets follows the optimal path under low traffic condition (MPIA=1.0). However as soon as the traffic increases (MPIA=0.005), the packets are stochastically switched to the other available paths as well.

5 Conclusion

The important contribution of the paper is a scalable analytical framework that is used to successfully model the behavior of *BeeHive*. The model obviates the need to do time-consuming simulations, sometimes lasting months, to study the scalability behavior of a routing protocol. The performance metrics estimated by our formal framework are approximately the same as obtained though network simulations. In future we want to undertake the following important tasks: (1) to do the scalability study in a similar fashion as was done by Farooq in [3] using time-consuming lengthy simulations, (2) extend the traffic model to session-oriented traffic also, (3) extend the network model to cater for heterogenous networks of varying bandwidth and link delays, and (4) extend the model to incorporate other Nature-inspired routing protocols like AntNet as well. The work in this phase will be the subject of forthcoming publications.

References

1. Bean, N., Costa, A.: An analytic modelling approach for network routing algorithms that use ”ant-like” mobile agents. Computer Networks 49(2), 243–268 (2005)
2. Di Caro, G., Dorigo, M.: Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research (JAIR) 9, 317–365 (1998)
3. Farooq, M.: Bee-inspired Protocol Engineering: from Nature to Networks. Natural Computing Series. Springer, Heidelberg (in press, 2008)
4. Farooq, M., Di Caro, G.: Routing protocols for next-generation intelligent networks inpired by colective behaviors of insect societies. In: Swarm Intelligence:Introduction and Applications. Natural Computing Series. Springer, Heidelberg (2008)
5. Kelly, F., Zachary, S., Zeidins, I.: Stochastic Networks Theorey and Applications. Oxford Science Publications (1996)
6. Taha, H.A.: Operations Research. John Wiley & Sons, Chichester (1982)
7. Trivedi, K.S.: Probability and Statistics with Reliability, Queuing and Computer Science Applications. John Wiley Interscience Publication, Chichester (2002)
8. Wedde, H.F., Farooq, M.: A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks. Journal of Systems Architecture 52(8-9), 461–484 (2006)
9. Wedde, H.F., Farooq, M.: A performance evaluation framework for nature inspired routing algorithms. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 136–146. Springer, Heidelberg (2005)
10. Wedde, H.F., Farooq, M., Zhang, Y.: BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 83–94. Springer, Heidelberg (2004)
11. Zahid, S., Shahzad, M., Ali, S.U., Farooq, M.: A comprehensive formal framework for analyzing the behavior of nature-inspired routing protocols. In: IEEE Congress on Evolutionary Computation, pp. 180–187 (September 2007)

A Grouping Genetic Algorithm Using Linear Linkage Encoding for Bin Packing

Özgür Ülker, Emin Erkan Korkmaz, and Ender Özcan*

Yeditepe University, Department of Computer Engineering, Kadikoy/Istanbul Turkey
{oulker,ekorkmaz,eozcan}@cse.yeditepe.edu.tr

Abstract. Linear Linkage Encoding (LLE) is a representation method proposed for grouping problems. It has already been used in solving data clustering, graph coloring and timetabling problems based on multi-objective genetic algorithms. In this study, this novel encoding scheme is investigated on bin packing again using a genetic algorithm. Bin packing benchmark problem instances are used to compare the performance of traditional recombination operators and custom made LLE crossover operators which are hybridized with parametrized placement heuristics. The results denote that LLE is a viable candidate for bin packing problem whenever appropriate genetic operators are chosen.

1 Introduction

Most of the *grouping problems*, such as, data clustering, graph coloring, or bin packing require partitioning of a set items into n mutually disjoint subsets [8]:

$$V = V_1 \cup ... \cup V_k \cup ... \cup V_l \cup ... \cup V_n \quad and \quad V_k \cap V_l = \emptyset, \quad where \quad k \neq l. \quad (1)$$

In different problems this partitioning process is subject to a different set of constraints. Various meta-heuristics such as simulated annealing [13], tabu search [10], genetic algorithms (GAs) [12], etc. have been applied to solve grouping problems. GAs derived from the population genetics and the Darwinian theory of evolution are powerful tools commonly used in search and optimization for solving complex problems ([4], [11], [12], [16]-[19]). In spite of the satisfactory performance of the traditional GAs on many NP-hard optimization problems, unfortunately, the same achievement is not usually observed on grouping problems.

This is because many evolutionary algorithms do not address the dynamics of a grouping problem: how to handle the groups. The commonly used representations usually suffer from redundancies due to the ordering of groups. Moreover the genetic material representing groups might easily be disrupted by the genetic operators and/or by the rectification process after the operators are applied. Therefore a genetic algorithm requires special operators for solving a grouping problem.

* The author is currently on leave of absence and working as a research fellow in the ASAP group at the University of Nottingham.

1.1 Bin Packing Problem

Bin packing is a combinatorial NP-hard grouping problem in which items of different sizes has to be packed into a minimal number of bins of fixed capacity C. In classical one dimensional bin packing problem [2], a sequence of $S = (I_1, I_2, ..., I_k, ...I_n)$ items, each with a size $s(I_k) \in (0, 1]$ are packed into a minimum number of unit-capacity bins (partition them into a minimum number m of subsets $B_1, B_2, ..., B_l, ...B_m$ such that $\sum_{I_k \in B_l} s(I_k) \leqslant 1, 1 \leqslant l \leqslant m$).

There are exact and approximate methods proposed for solving bin packing. The exact methods guarantee that the optimal result will be achieved, while the approximate approaches provide $satisfactory$ approximations. Martello and Toth's branch-and-bound reduction algorithm [15] (MTP) is an exact approach which is used as the basic reference in most comparative studies of bin packing. MTP although slow (for large instances) generally gives excellent results. The MTP procedure attempts to find bins dominating all others. After such a bin is found, the problem is reduced by removing the dominating bin. In order to prevent an exponential search, only dominating bins of at most three items are taken into account.

Constructive heuristics and (meta-)heuristics constitute approximate approaches. In *the first fit heuristic* (FF), an item I_k is placed in the first (lowest indexed) partially-filled bin B_j into which it could fit ($capacity(B_l + s(I_k) \leqslant 1$). If this is not possible, a new bin containing I_k as the first item is created. A variant of first fit is *first fit decreasing* (FFD) in which items are first sorted in decreasing weight and then items are picked up one by one beginning with the largest item and each item is placed into the first bin that can accommodate it.

In the best fit heuristic, an item I_k is placed in the partially filled bin B_l with the *highest* level ($level(B_l) \leqslant 1 - s(I_k)$) and ties are broken in favor of lower index bins. Similar to FF, Best Fit has a decreasing variant, in which items are again sorted in decreasing order and placed into the best-filled bin that can accommodate it. Although best-fit decreasing is slightly more complicated than FFD, surprisingly it cannot beat FFD. Both heuristics have the worst case performance of $\frac{11}{9}Opt + 4$ where Opt is the number of bins in the optimal solution [2]. Dosa [5] proved that the tight bound for FFD is $\frac{11}{9}Opt + \frac{6}{9}$.

Falkenauer [7] uses a Hybrid Grouping Genetic Algorithm (HGGA) which is heavily modified to suit the structure of the grouping problems. His genetic algorithm works with whole bins rather than with individual items. In HGGA representation, a standard chromosome representing the IDs of the items are augmented with a group part, encoding the groups on a one gene for one group basis. The important point with the genetic operators is that they work on the group part of the chromosome, the standard item part is just used to identify which items form which group. Falkenauer used a strategy similar to the domination criterion of Martello and Toth to place the eliminated bins (free items). Free items are swapped with non-free items (items currently placed within bins) such that the bins will consist of few large items rather than many small items. Items that cannot be placed with this replacement strategy are re-inserted into the solution using a first-fit heuristic. Mutation works also similarly; it destroys a few bins from the element and reinserts the missing items using the mentioned local search procedure.

Linear Linkage Encoding (LLE) is proposed as a novel representation scheme for grouping problems [6]. LLE uses a link-based structure for objects within the same group. Genetic operators work on the encodings by altering the links. It is reported that the performance of LLE is superior to *number encoding* (NE) which is the most common encoding scheme used in grouping problems. Unlike NE, LLE does not require an explicit bound on the number of groups that can be represented in a fixed-length encoding. The greatest strength of LLE is that the search space is reduced considerably. There is a one to one correspondence between the encodings and the solutions when LLE is used. Du et al. [6] utilized LLE for data clustering, while Ulker et al. [21] experimented with LLE on graph coloring and timetabling problems. A multi-objective genetic algorithm is used in both studies. In this paper, these previous studies on LLE is extended. Bin packing is chosen as a testbed for investigating LLE further using a single objective genetic algorithm. The performance of different genetic operators are compared based on LLE.

2 Linear Linkage Encoding for Grouping Problems

Number Encoding (NE) is the most widely used representation in grouping problems. In this scheme, each gene is reserved for an object and the value of the gene indicates the group ID that the corresponding object belongs to. If six objects are to be grouped, the individual 234212 is a valid chromosome in NE and it encodes the solution where first object is in group 2, second in 3 and so on. Other chromosomes exist that represent exactly the same solution. 123141 is such an example where the order of the groups is different. However, this ordering proposed by the representation is irrelevant in terms of building the solution. In [20], it is denoted that NE is against the minimal redundancy principles for encoding scheme.

Group Encoding (GE) is an alternative representation for grouping problems. In this scheme, the objects in the same group are placed into the same partition set. The sets are separated from each other by using special markers in the chromosome. For instance, the above example can be represented as $(1, 4, 6)(2)(3)(5)$ in GE. However, the ordering redundancy still holds. For instance, $(2)(3)(5)(1, 4, 6)$ would represent the same solution.

A special representation is used in [9] for the problem. This representation augments standard number encoding with a group part. However, the same redundancy problem exists for this augmented form, too. The difference from group encoding is that traditional search operators work on the group part of the encoding.

The redundancy that exists in above representational schemes is due to the symmetric structure of search spaces in grouping problems. Many researchers have proposed symmetry breaking methods to prune redundant search spaces [3] [1]. Linear Linkage Encoding (LLE) is a new representation scheme proposed to eliminate the symmetry problem in grouping problems. The encoding scheme [6] (LLE) was first proposed to solve the clustering problem. A multi-objective genetic algorithm (MOGA) is used in this application. In [6], it is denoted that LLE can reduce the search space considerably.

In LLE, a gene is reserved for each object just like NE. However, the value of a gene is interpreted as a link from one object to another object of the same group. With n objects, any partition set on them can be represented using a chromosome of length n. Two objects are in the same group if either one can be reached from another through the links. When a gene value is equal to its own index, then it is considered as the last element of a group (ending node). The following two constraints are imposed on LLE links:

- The value of each gene is greater than or equal to its index but less than or equal to n.
- No two genes can have the same value except if one of them is an ending node.

In LLE, the objects of a group form a linear path ending with a self referencing last item. In [14], it has been proved that a one to one mapping has been obtained between the possible partitions and the chromosomes of LLE (Figure 1). The genetic operators may disturb the two constraints mentioned above. A rectification process would be needed in order to recover the chromosome in such a case. The process is quite straightforward and the details are presented in [6].

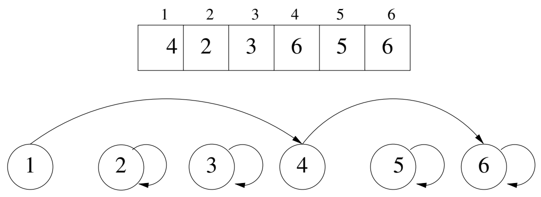


Fig. 1. A LLE chromosome and the partition it represents

The non-redundancy advantage of LLE diminishes if a crossover operator causes huge jumps on the search space. Traditional crossovers like 1PTX or UX can easily destroy the building blocks and hence result in huge changes on the partition represented. It is important to preserve the order of the groups as much as possible during a genetic operation. Therefore, two different ordering mechanisms which assign group IDs to the groups are investigated within the context of LLE. These ordering mechanisms are based on the cardinality of the groups and the lowest index number in each group. In Cardinality Based Ordering, the group ID is determined based on the group cardinality (set size). The group with the highest cardinality is assigned group ID 1, the second highest will be identified as group 2, and so on. In Lowest Index Ordering, the IDs are assigned to groups based on the the smallest index in each group.

3 A Grouping Genetic Algorithm for Bin Packing Problem

The GA used in this study utilizes LLE as the representation scheme. The individuals are initialized using the FF heuristic and the resulting individuals are converted into the LLE form. In order not to have the same individual for the whole population, a random permutation of the items are fed into the FF heuristic. At the end of the initialization it is ensured that the capacity of each bin is not exceeded. As a mate selection method, tournament selection is used. Some preliminary experiments are performed and it is observed that using higher selection pressure yields better results. The performances of different crossover operators, including the traditional ones (1PTX, UX) are compared. A non-traditional smart mutation is utilized. A trans-generational GA replaces the current generation with the next one by keeping only the best individual from the current population.

A straightforward fitness function would just take the inverse of the number of bins. However, as pointed by Falkenauer [9] as well, such a fitness function will result a very unfriendly fitness landscape in which many combinations with one more bin than optimal solution will have the same fitness value. Instead, the function proposed by [9] is used in this study:

$$f(s) = \frac{\sum_{i=1}^{N}(F_i/C)^2}{N} \tag{2}$$

where, N is the number of bins, F_i is the fill of bin i and C is maximum bin capacity for a given solution s.

3.1 Crossover Operators

Ulker et al. [21] reports that Lowest Index Max Crossover (LIMX) performs well on different graph coloring and timetabling problems. In LIMX, a single child is generated using two individuals with two aims: transmit large groups to preserve Cardinality Based Ordering, and to transmit groups beginning with lowest index number (to preserve Lowest Index Ordering). Beginning with the lowest index number (vertex) which has not been assigned first we calculate the length of the links (path length) in both parents. Already assigned vertices are not counted in this link length calculation. This allows finding the largest set in parents beginning with the lowest index number. Then the links (and thus vertices) are transmitted to the child from the parent based on the link-lengths. Then, the next unassigned lowest index number is found and the process is repeated until all vertices are assigned. For example, assume that $\{(1,3,6),(2,4),(5)\}$ and $\{(1,2),(3,4,5,6)\}$ are selected as mates. The starting lowest index is 1. $(1,3,6)$ is longer than $(1,2)$, so $(1,3,6)$ is copied to the child. Then, the indices $(1,3,6)$ are deleted from the mates: $\{(2,4),(5)\}$ and $\{(2),(4,5)\}$. Now, the current lowest index is 2. $(2,4)$ is larger than (2) so it is transmitted to the child and deleted from the mates: (5) and (5). Finally, (5) is copied to the child as the last group and a new offspring is generated as $\{(1,3,6),(2,4),(5)\}$.

A modified uniform crossover (MUX) method is utilized on the clustering problem with LLE in [14]. In MUX, instead of the actual values like in UX, the value of the ending node of the group in which the item belongs to is passed to the offspring. This ensures groups were not separated. It is observed that this crossover tends to combine the groups having the same ending node in both parents. MUX is the second operator tested in bin packing domain.

3.2 Smart Mutation

A smart mutation that targets no overfilled bins during the packing is utilized within the grouping GA. In mutation, k randomly chosen bins are destroyed and the contents of these bins are redistributed to rest of the bins. The performance of GA is observed for various values of the parameter k, referred to as *mutation rate*. Furthermore, due to the nature of the crossover operators, the resulting individual may have bins whose capacities are exceeded and thus may need an additional repair procedure apart from the usual LLE rectification mechanism. This repair procedure checks all of the bins and removes randomly selected items from the over-filled bins until the capacity is not exceeded anymore. These removed items are combined with the items from the previously deleted bins. Then, all of them are reinserted into the rest of the bins using a heuristic. If it is not possible to insert some items without exceeding the capacity of existing bins, then a new bin is created. FF and FFD heuristics are used as repair heuristics.

Like previous algorithms of Falkenauer [7] and reduction algorithm of Martello and Toth [15], a procedure based on the domination criterion has been adopted. When an excess item is to be inserted back to the solution, first it is compared with the items already present in the bins. An excess item replaces an item in the bin while not causing an overfill in the bin and the replaced item is put into the excess items list. This procedure increases the number of well-filled bins. In the first-fit heuristic, the lowest-index based ordering is used. Due to the nature of the bin packing instances, cardinality based ordering will be no different than random ordering, therefore it is ignored.

4 Experimental Results

Two sets of test instances provided by Falkeanuer [7] are used in the experiments. In the first set, the maximum bin capacity is set to 150 and each integer item is randomly generated using a uniform distribution between 20 and 100. Falkenauer reports that this distribution gives most difficult instances for the method proposed by Martello and Toth [15]. In the second set, the item sizes are drawn from the range $(0.25, 0.50)$ to be packed into bins with maximum capacity 1. In these instances, a well-filled bin must contain one large item and two small items. That is why Falkenauer referred them as 'triplets'. Falkenauer [7] points out a similarity between triplets and $3SAT$ which is considered as the most difficult $kSAT$ problem. In order to preserve the difficulty of the problem, the generated instances have known local optima with maximum bin capacities of 1000. 20 different instances of triplets containing 60, 120, 249 and 501 items are then generated.

The experiments are performed on Pentium IV, 2 GHz Linux machines with 256 Mb memory. The population size, tournament size and crossover rate are fixed as 100, 10 and 1.00, respectively. The first fit (FF) and first fit decreasing (FFD) heuristics are used as repair mechanisms. Mutation rate is the number of bins destroyed in an element. Different experiments destroying $1, 2$ and 4 bins are carried out. For consistency with the previous experiments of Falkenauer [7] and Martello [15], each instance in a set is tested only once. Hence, as a total of twenty runs are performed.

The performances of four crossover operators (1PTX, LIMX, UX and MUX) are tested using FF and FFD with mutation rates of $1, 2$ and 4 providing a total of 6 combinations for each crossover. In order to compare in a fixed heuristic and mutation rate setting, a ranking mechanism that takes ties into account is implemented. The ranking method takes into consideration the success ratio (number of times the optimal solution is found), the mean number of bins found and the mean number of generations. The t-tests are also carried out to ascertain if there are statistically significant differences between different settings. If one mutation instance (combination of repair heuristic and mutation rate) is better than all the others then it is given a ranking of 1, and if it is worse than all others, it is given a ranking of 6. The average ranking of each mutation instance over the problems for each crossover is presented in Figure 2. It is observed that crossover performance differs based on the repair heuristic utilized. Although, there is no significant performance variance among the choice of repair heuristic and the mutation rate, considering the overall performance of each, the best repair heuristic turns out to be FF. 1PTX performed best in a FF and the mutation in which 2 bins are destroyed (FF2). LIMX operated best when used with FF1 which is closely followed by FFD4. For UX, FFD2 performed best while for MUX FFD1 is best and closely followed by FFD2.

Table 1 provides a comparison of the results for different approaches, including the grouping GA with the best setups for each crossover (1PTX - FF2, LIMX - FF1, UX - FFD2, MUX - FFD1). The results clearly show that the proposed algorithm is superior to Martello and Toth's [15] reduction algorithm in all problem

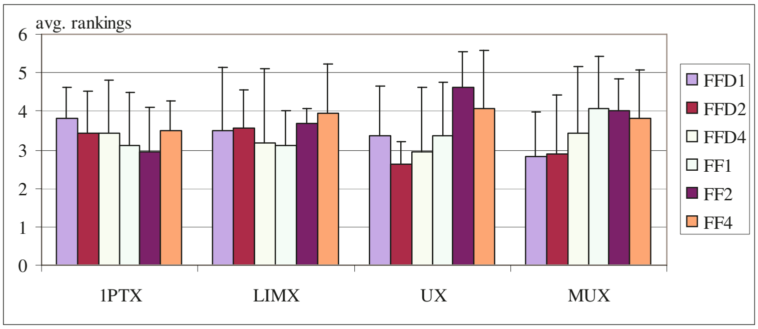


Fig. 2. Performance comparison of crossovers using different repair heuristics

Table 1. Mean number of bins obtained in best setups for each crossover, *Theo* denotes theoretical minimum lower bound on the number of bins

Instance Set	Theo	1PTX FF2	LIMX FF1	UX FFD2	MUX FFD1	HGGA [7]	M&T [15]
U120	49.05	49.05	49.10	49.05	49.05	49.15	49.15
U250	101.55	101.70	101.80	101.75	101.65	101.70	102.15
U500	201.20	201.30	202.35	201.50	201.30	201.20	203.40
U1000	400.55	400.65	417.50	401.45	401.05	400.55	404.45
T60	20.00	20.95	21.00	21.00	21.00	20.10	21.55
T120	40.00	41.00	41.00	41.00	41.00	40.00	44.10
T249	83.00	84.00	84.00	84.15	84.05	83.00	90.45
T501	167.00	168.00	168.85	169.80	169.20	167.00	181.85
AVG	132.79	133.33	135.70	133.71	133.54	132.84	159.64

sets. Falkenauer's HGGA [7] remains however the best algorithm in terms of overall solution quality especially for the more difficult triplet instances. However, for the uniform distribution instances, LLE with 1PTX is very competitive, in fact for smaller instances it provides some packings that cannot be found with HGGA. For the triplet instances, LLE with 1PTX is consistently one bin short of the optimal packing.

The results presented in Table 1 are also evaluated using some statistical tests. Each approach (1PTX - FF2, HGGA, M&T, etc.) is ranked according to its performance for each problem instance. The ANOVA test over the average ranks shows that performances of 1PTX - FF2, HGGA and M&T are not equal to each other within a confidence interval of of 99.99%. Then, t-tests are used to make pairwise comparisons between 1PTX - FF2 and the others. It has been observed that 1PTX - FF2 and HGGA generate significantly better performances as compared to M&T within a confidence interval of of 99.99%. HGGA seems to perform better as compared to 1PTX - FF2, however, this difference in performance is not statistically significant under the t-test.

In terms of mean number of bins and number of generations, 1PTX usually provides the best results especially for difficult large triplet instances. LIMX performs somewhat inconsistently. It sometimes gives results close to 1PTX on some large triplet instances. However it lags far behind especially in the larger uniform distribution instances. UX and MUX behave consistently and yield an acceptable performance in all test setups. LIMX and the other crossover operators are not particularly suitable for the bin packing instances tested because for all of the instances average number of items per group is quite small (2 to 3 for uniform distribution instances and 3 for triplet instances). Preservation of large groups is not important due to the small average size of the bins and the low epistasis between items. Because of small and low epistasis bins, 1PTX has a good performance as the likelihood of destruction of well-filled bins is lower.

5 Conclusions

In this study, the performance of LLE has been tested on bin packing. Several crossover operators that can be used with LLE have been investigated. Unlike graph coloring [21], the performance of the traditional crossovers shines when used in bin packing. They were able to generate very competitive results close to hybrid grouping genetic algorithm (HGGA) of Falkenauer. They match HGGA in uniform distribution instances and for smaller instances outperform them. However for most difficult triplet instances, 1PTX, the best performing crossover in the test setup, was constantly one bin short of the optimal solution. This problem probably requires enhancements in the mutation operator and this is one of the future research directions.

Crossovers utilized in graph coloring domain (e.g., LIMX) cannot perform competitively on bin packing instances. This is an expected result since in bin packing it is not crucial to preserve very large sets during generations. The most difficult test instances usually require packing few items to a bin (2 to 3 in this case). This is why ordering of groups based on cardinality does not make sense as it will essentially be no different than random ordering. Hence, traditional crossover operators perform quite well in this domain.

Linear linkage encoding is a viable candidate for solving grouping problems especially if the number of groups is not known beforehand. In such problems, the search is performed on a smaller search space than other encodings such as number encoding. New operators that will make better use of this representation awaits research. Moreover, other approaches for solving grouping problems may also utilize LLE as their representation method, such as, hyper-heuristics [18].

Acknowledgments. This research is funded by TUBITAK (The Scientific and Technological Research Council of Turkey) under the grant number 105E027.

References

1. Backofen, R., Will, S.: Excluding symmetries in constraint-based search. Principles and Practice of Constraint Programming, 73–87 (1999)
2. Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: a survey., 46–93 (1997)
3. Crawford, J., Ginsberg, M.L., Luck, E., Roy, A.: Symmetry-breaking predicates for search problems. In: Aiello, L.C., Doyle, J., Shapiro, S. (eds.) KR 1996: Principles of Knowledge Representation and Reasoning, pp. 148–159. Morgan Kaufmann, San Francisco (1996)
4. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
5. Dosa, G.: The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is FFD (I) $\leq$ 11/9 OPT (I). In: Chen, B., Paterson, M., Zhang, G. (eds.) ESCAPE 2007. LNCS, vol. 4614, pp. 1–11. Springer, Heidelberg (2007)
6. Du, J., Korkmaz, E., Alhajj, R., Barker, K.: Novel clustering approach that employs genetic algorithm with new representation scheme and multiple objectives. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, Springer, Heidelberg (2004)
7. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics 2, 5–30 (1996)

8. Falkenauer, E.: Genetic Algorithms and Grouping Problems. John Wiley and Sons, Chichester (1998)
9. Falkenauer, E., Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: Proc. of the IEEE 1992 Int. Conf. on Robotics and Automation, Nice, France, pp. 1186–1192 (1992)
10. Glover, F.: Tabu search, part 1. ORSA Journal on Computing 1, 190–206 (1989)
11. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
12. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
13. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
14. Korkmaz, E.E., Du, J., Alhajj, R., Barker, K.: Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. Intelligent Data Analysis 10(2), 163–182 (2006)
15. Martello, S., Toth, P.: Lower bounds and reduction procedures for the bin packing problem. Discrete Applied Mathematic 28(1), 59–70 (1990)
16. Ozcan, E.: Memes, Self-generation and Nurse Rostering. In: PATAT 2006. LNCS, vol. 3867, pp. 85–104. Springer, Heidelberg (2007)
17. Ozcan, E., Alkan, A.: A Memetic Algorithm for Solving a Timetabling Problem: An Incremental Strategy. In: Baptiste, P., Kendall, G., Kordon, A.M., Sourd, F. (eds.) Proc. of the 3rd Multidisciplinary Int. Conf. On Scheduling: Theory and Applications, pp. 394–401 (2007)
18. Ozcan, E., Bilgin, B., Korkmaz, E.E.: A Comprehensive Analysis of Hyperheuristics. Intelligent Data Analysis 12(1), 3–23 (2008)
19. Ozcan, E., Onbasioglu, E.: Memetic Algorithms for Parallel Code Optimization. International Journal of Parallel Programming 35(1), 33–61 (2007)
20. Radcliffe, N.J.: Formal analysis and random respectful recombination. In: Belew, R.K., Booker, L.B. (eds.) Proc. of the 4th Int. Conf. on GAs, pp. 222–229 (1991)
21. Ulker, O., Ozcan, E., Korkmaz, E.E.: Linear Linkage Encoding in Grouping Problems: Applications on Graph Coloring and Timetabling. In: Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling, pp. 303–319 (2006)

Optimization of Feature Processing Chain in Music Classification by Evolution Strategies

Igor Vatolkin[1] and Wolfgang Theimer[2]

[1] TU Dortmund, Germany
Igor.Vatolkin@cs.tu-dortmund.de
[2] Research in Motion, Bochum, Germany
Wolfgang.Theimer@ieee.org

Abstract. In this paper a new method based on evolution strategies (ES) is presented to optimize a classifier for personal music categories. The user assigns songs to multiple personal music categories: Examples from each category are selected in order to train a category-specific classifier using musical features as input. The classifier then ranks all songs according to their similarity to the category examples. Since an exhaustive search for parameters maximizing the classifier performance is not feasible an ES is applied. The experiments show a significant performance increase for various music categories due to the ES optimization.

1 Introduction

The music market has changed profoundly during the past few years. The digital distribution of music is gradually taking over the dominant position from the sales of physical media such as music CDs and DVDs. Furthermore, the capabilities of consumer devices like portable media players and (smart)phones with music playback are increasing constantly. This leads to the situation that a user can now carry his or her complete music collection in a mobile device.

For music playback the form factor of the device is not important: The audio quality even in very small terminals can be excellent. But the small physical size of a device limits the user interaction due to the restricted size of a keypad or display. It is very time consuming and suboptimal to manage *textual menu lists* such as artist or track lists with thousands of entries on a device the size of your palm. This is one motivation for our work to organize large music collections more intuitively. Another observation is that users have a very *personal* taste of music. The pre-defined music categories such as genres or sub-genres might help, but do not take personal listening preferences into account. We argue in the following that music listeners should be able to define their *personal categories* of music by giving examples and let an intelligent music classification system rank the complete database according to these user preferences.

In our approach we investigate how far the music taste of a person can be approximated from the audio content of a small number of music examples, prototypes that perfectly match a personal category and also counter-examples that are the complete "opposite" of this personal music category. In the following section about previous work we outline how the music content is analyzed: A large collection of music features is extracted and processed to achieve a more compact representation. The resulting feature vectors are classified and a summary rating indicates how similar each music track is to a user-defined music category.

This leads to the actual problem we want to address with this paper: The music processing chain is complex and produces a huge amount of intermediate data when a large music database is analyzed. There is no analytical solution available that provides the optimal parameter settings for the processing chain. Since the number of parameters is high an exhaustive search for each parameter combination is not feasible. This topic is outlined in more details in problem statement section and our solution to this problem based on evolution strategies for the categorization is described in the following section.

In the experiment section we demonstrate the effectiveness of the evolutionary approach in the context of training an optimal classifier for personal music categories and show the results of the optimization. We argue that this optimization is effective and especially valid in our situation where we cannot provide an analytic solution to come up with the optimal parameters.

Finally we give a conclusion of our work and point to promising open research questions which should be addressed in future.

2 Processing Chain and Previous Work

2.1 Feature Extraction

A set of musical features is extracted from all music tracks. The N raw features $f_i, i \in [1, N]$ describe different characteristics of music, i.e., timbre, harmony, melody, rhythm, time and structural properties of music [21]. We used a set of 33 features, the mathematical definitions can be found in [19]. They are extracted in a sequence of M time windows for one track of music. The set of computed features for one time window forms a raw (column) feature vector $\mathbf{f} = (f_1, f_2, ...f_N)^T$. Concatenation of the feature vectors for each time window creates a N x M feature matrix $\mathbf{F}$, where the number of columns is equal to the number of time windows (see Eq. 1):

$$\mathbf{F} = \begin{pmatrix} f_1(t_1) & f_1(t_2) & \dots & f_1(t_M) \\ f_2(t_1) & f_2(t_2) & \dots & f_2(t_M) \\ \vdots & \vdots & \ddots & \vdots \\ f_N(t_1) & f_N(t_2) & \dots & f_N(t_M) \end{pmatrix} \tag{1}$$

The timbre features and some harmony / melody features are computed for 23 ms time windows (512 samples at a sampling rate of 22.05 kHz) with no overlap. But there are also more complex features that require longer time windows for the computation, like e.g., the fundamental frequency / pitch estimation or the melody analysis. Rhythm features are analyzed in time windows of several seconds providing sufficient signal statistics for a reliable estimate. Structural features of music are not yet taken into account, but typically are computed in even longer time intervals (in the order of 30 s) [9]. For details about the mathematical definitions of features refer to [10], [15], [20], [16].

2.2 Processing of Features

After the extraction of raw features the created feature matrix **F** must be converted to one ore more classifier input vectors. Since the ranges of feature values can be very different, normalization is often applied before classification (e.g. mapping of features to the intervals [-1,1] or [0,1]).

Methods operating on rows of the feature matrix normally seek to reduce the number of features. Principal component analysis [17] and linear discriminant analysis [7] are statistical methods to reduce the data dimensionality. In correlation-based feature selection highly correlated features are not submitted to the classifier. One can start with an empty group of features adding one-by-one a new feature which is least correlated with the existing feature set. Another possibility is to start with the full set of features and to take away the mostly correlated feature one-by-one until the desired number of features is reached [6], [5]. Sequential forward selection starts with an empty group of features identifying new features best improving classification performance in each iteration step. Frequently new features are created by calculating derivatives or running means of existing ones [1].

Another group of methods operates on the columns of the feature matrix. The information of tatum and beat times in a music piece can be used for subsampling the feature matrix. Tatum times measure the duration of the shortest note. We describe further the tatum reduction method with $R_{tat} = \{n, t, b\}$. If no tatum reduction is applied, $R_{tat} := n$. If it is applied, only the features from time windows with tatum times ($R_{tat} := t$) or exactly in the middle between tatum times ($R_{tat} := b$) are saved for further classification since notes are changing on that time scale. Also partitions can be built dividing a track in parts of equal length (e.g. 5 seconds). Each partition provides a single classifier input vector. If some structural information about a track is available, features from different song parts can be processed independently: It will be possible e.g. to identify feature values characteristic for the track from song intro, refrain and bridge.

Finally, the pruned feature matrix is converted to one or several classifier input vectors, labeled with ground truth information by the user. In our experiments we use a first-order Gaussian model, saving mean value and standard deviation over time from each feature. Alternatively more complex Gaussian models using larger number of Gaussian bell-shaped curves or methods which calculate the optimum number of Gaussians can be applied [13]. Statistical methods can compress a sequence of feature values, e.g. by creating histograms or autoregressive moving average models [2].

2.3 Classification of Music Categories

The ultimate target of our system is that the music classifier learns the user's musical taste and assigns the same similarity values to music tracks as the human user. In order to train the classifier and test the performance different users are asked to rank 1139 music tracks (our music database) by similarities to their personal music category (reference input). Each user has to identify 10 music tracks as prototypes for one personal music category i (similarity $s_{i,ref}(n) = 1$) and 10 tracks as counter-examples of the music category (similarity $s_{i,ref}(n) = 0$). These 20 songs are used for training. The user also ranks the remaining 1119 tracks of music by assigning one of four discrete similarity values to them: $s_{i,ref}(n) = 1$ - music fits perfectly to music category (but is not used as positive example for training), $s_{i,ref}(n) = \frac{2}{3}$ - good, but not perfect fit with music category, $s_{i,ref}(n) = \frac{1}{3}$ - weak similarity with category, $s_{i,ref}(n) = 0$ - music does not fit at all to personal music category (but is not used as counter-example for training). We used an even number of similarity values in order to not allow users to choose an average rating thus forcing them to make a decision. Providing finer or even continuous ranking values easily leads to confusion: People hardly remember their previous choice for similar songs.

The output of the classifier $s_i(n)$ is compared to $s_{i,ref}(n)$. An optimal classifier would perfectly approximate the reference similarities and the deviations between $s_{i,ref}(n)$ and $s_i(n)$ would be zero for each music track n. In reality there are differences and as measure for the classifier performance the mean squared error divided by the maximum possible mean squared error is chosen (L is the song number):

$$E^2 = \frac{1}{E^2_{max}} \; \frac{1}{L} \sum_{i=1}^{L} \left(s_i(n) - s_{i,ref}(n)\right)^2 \qquad (2)$$

$$\text{with } \; E^2_{max} = \frac{1}{L} \sum_{i=1}^{L} e^2_{i,max}(n) \; \text{ and}$$

$$e_{i,max}(n) = \begin{cases} 1 & : \quad s_{i,ref}(n) = 0 \vee s_{i,ref}(n) = 1 \\ \frac{2}{3} & : \quad s_{i,ref}(n) = \frac{1}{3} \vee s_{i,ref}(n) = \frac{2}{3} \end{cases}$$

The normalization of the error by the divisor E^2_{max} is needed since the maximum possible error with regard to a value $s_{i,ref}(n)$ is $\max(s_{i,ref}(n), 1 - s_{i,ref}(n))$. In section 5 the E^2 measure is applied to assess the classifier performance in different experiments.

The purpose of classification algorithms is to relate music songs to given categories, allowing each song to be member of several categories. The results of independent classifiers for different sets of features can be combined [4]. Many classification strategies were developed for data mining. In our work we use supervised learning techniques for music classification, which learn from given labeled data (ground truth). For details and an overview of classifiers, see [3].

The *divide-and-conquer* algorithm builds decision trees which consider only one most important feature in each node. The importance of every feature is estimated on the basis of an entropy measure. Several enhancements of divide-and-conquer lead to an algorithm named C4.5 [12]. It typically constructs very compact decision trees. The experiments described later deploy C4.5 decision trees.

3 Problem Statement

In the last section we have outlined the complete music processing chain. For a given classifier algorithm the input parameters, i.e. the set of features and the duration of a partition, are fixed. Based on this input the classifier performs a training to maximize the recognition rate.

But the classifier can only achieve a *local* performance optimum because it is not proven that the specific input parameter set is the best one. Since an analytical solution to determine the best input parameter set is not available, an exhaustive search for all possible input parameter combinations would be needed to find the *global* optimum. It is not feasible to perform this exhaustive search by e.g. executing a grid-based search due to the high number of the input parameters: We have 33 features or feature groups that could switched on or off in the feature extraction phase. In addition the duration of the partition is a continuous number between a minimum length (here assumed to be 0.5 s) and a maximum duration (set to 30 s). Even if we quantize the partition length and only evaluate some discrete durations between the minimum and the maximum value the number of combinations is very large.

Therefore we propose an alternative approach: We use an evolution strategy on top of a classifier to determine a specific set of input parameters and feed it into the classifier. During the training phase the music examples for the personal music categories are used as before and a certain classifier performance is the result. We use the mean squared error between the user ratings (ground truth) and the classifier results as the fitness criterion. This fitness value is fed back to the evolution strategy to determine a new input parameter set for the classifier. The best input parameter set so far is always stored and compared to the new parameter set. This approach avoids the complete computation of all input parameter combinations, but might converge to a local instead of the global optimum. The details are presented in the next section.

4 Evolution Strategies for Music Categorization

Evolution strategies have not yet been widely applied to the optimization of the feature processing chain in music information retrieval. A very generic method was investigated in [8]. In contrast, we wanted to analyze certain processing parameters and measure their influence on classification quality. We considered at first to let the ES operate on (1) the feature set for processing and classification and (2) the size of a partition T_p, which corresponds to one classification sample ($T_p \in [500, 30000]$). The smallest possible change in partition size (mutation step size) was set to 1 ms. Thus, with a complete number of features $N = 33$, the number of all possible processing combinations is $2^{33} \cdot 25501$.

The representation of an individual consists of a binary vector $\mathbf{m}$ with $m_i \in \{0, 1\}$ indicating if a feature i is used (1) or not (0) and an integer value $T_p \in [500, 30000]$. So the optimization problem can be described as

$$\text{minimize } E^2(\mathbf{m}, T_p) \text{ with restriction } T_p \in [500, 30000]\,. \qquad (3)$$

We use a standard (1+1)-ES with random individual initialization. The number of iterations is set to 100 generations. Two mutations are applied per generation. The first mutation flips a bit in $\mathbf{m}$ with a probability $1/N$, switching on/off the processing of the corresponding feature. The second mutation changes T_p and calculates the step size with the method for integer mutation proposed in [14]. However the expected mean step size depends on the current generation G and is calculated from

$$S = 15000 \cdot 0.96^G + \frac{1 - 15000 \cdot 0.96^G}{100} \cdot G. \qquad (4)$$

For the first generation ($G = 0$) the expected mean step size is 15000 and is equal to half of the complete interval length. For the last generation ($G = 100$) the expected mean step size is 1, so the ES makes larger steps changing partition size at the beginning and reduces them to the expected minimal step size during the last generations. We also consider to use self-adaptation for step size calculation. However no comparable experiences exist and these experiments are planned for the near future.

5 Experiments

For experiments we used as a basis three personal music categories from [18]. 1139 songs were classified, with 10 positive and 10 negative prototypes. However, since the feature processing of all music songs during one fitness evaluation of ES is a very time-intensive procedure, the test set was reduced to 176 music tracks. From each CD two tracks were chosen randomly, so the validation results of this song subset are representative, despite of the smaller number of tracks.

For the comparison with ES optimization, we extended the grid-search experiments from [18], using feature reduction between tatum windows and a partition size of 2.5 seconds. We used the C4.5 decision tree algorithm for classification. The results are summarized in Table 1. Experiments with $R_{tat} = b$ are almost always better than with $R_{tat} = t$. The stability of sound between tatum times can explain this observation. In comparison to $R_{tat} = n$, processing of tatum windows slightly reduces the performance, and the processing of windows between tatum times slightly improves it. At any rate the tatum reduction saves about 90% computation time and storage space, since the features from the complete song are not required any more. Another observation is related to the partition size: Smaller partitions often provide better classification results. Too large partitions mix a lot of different data from the song, but too small partitions can be also dangerous since they can lead to overfitting of classifiers.

Table 1. E^2 for classifier results without optimization, evaluated on three categories (ED, IV, WT) with different tatum reduction methods

	Personal music category								
	ED			IV			WT		
T_p	R_{tat}								
	n	t	b	n	t	b	n	t	b
T_{song}	0.5740	0.6220	0.5796	0.2600	0.2253	0.2600	0.0835	0.0867	0.2174
10 s	0.2496	0.2655	0.2181	0.1029	0.1357	0.1050	0.0755	0.0748	0.0752
5 s	0.2309	0.2459	0.2293	0.0982	0.1168	0.1083	0.0636	0.0663	0.0647
2.5 s	0.2339	0.2311	0.2250	0.1015	0.1162	0.1024	0.0635	0.0654	0.0623

After the experiments with a fixed combination of processing characteristics, we started ES to find the best partition size and feature set. For each of the three user categories and $R_{tat} = \{n, t, b\}$ we started 10 runs. Table 2 contains E^2 mean values for these runs after 100 generations and lists the best individuals found during optimization with ES. Figure 1 depicts for a one category (here WT) mean E^2 values and 95 percent confidence intervals decreasing from generation 1 to 100.

The runs with ES outperformed the experiments without optimization. In regard to the classification performance, no tatum reduction method can be guaranteed to be the first choice. However, saving of features only from windows with tatum times or between tatum times saves disc space and processing time while resulting in a similar performance.

Another interesting observation is the partition size. Figure 2 shows all parent individual fitness values dependent on the partition sizes. These plots and Table 2 underscore that no general strategy for partition size choice can be recommended. The first category optimization (ED) leads to very small partition sizes (around 1 s), and the third (WT) to larger partition sizes (12-24 s). The possible assumption is that the more complex categories (like ED, where

Table 2. Mean E^2 (1st row) over 10 runs, best E^2 (2nd row), best T_p (3rd row) and best N_{best} (number of used features) for classifier results with optimization by (1+1)-ES, evaluated on three categories (ED, IV, WT) with different tatum reduction methods

	Personal music category								
	ED			IV			WT		
	R_{tat}								
	n	t	b	n	t	b	n	t	b
$\widetilde{E^2}$	0.1935	0.2037	0.1856	0.0703	0.0817	0.0738	0.0524	0.0509	0.0527
${E^2}_{best}$	0.1819	0.1826	0.1761	0.0654	0.0733	0.0676	0.0469	0.0469	0.0472
$T_{p_{best}}$	0.931 s	1.150 s	1.375 s	0.525 s	0.967 s	4.041 s	12.023 s	19.868 s	24.080 s
N_{best}	14	15	15	15	12	16	25	16	13

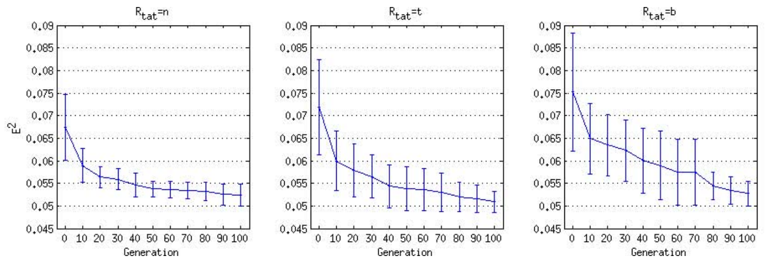


Fig. 1. Mean E^2 values and 95 percent confidence intervals for each tenth generation during the optimization of category WT

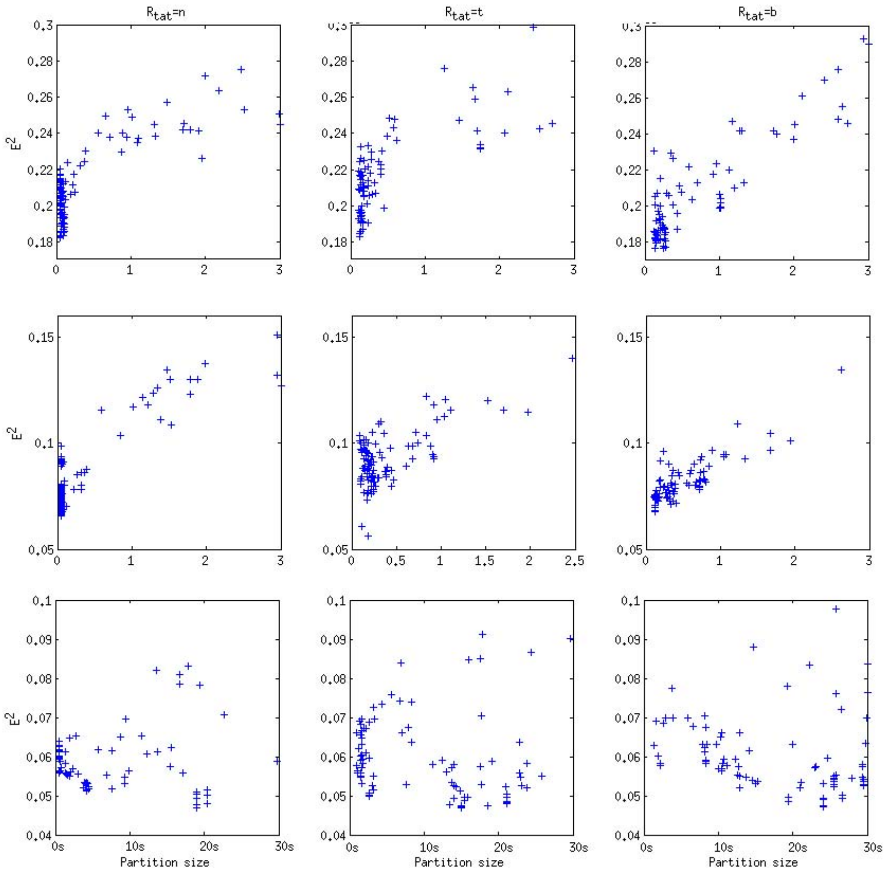


Fig. 2. E^2 values (all parent individuals of all iterations) depending on partition size (top row: category ED, middle row: category IV, bottom row: category WT)

the best E^2 found was equal to 0.1761) can be better categorized with smaller partitions, and the more simpler categories should be categorized with larger partitions. Here further investigations are required. Also, if we compare the different tatum reduction methods, it holds for all three categories, that $T_p(R_{tat} = n) < T_p(R_{tat} = t) < T_p(R_{tat} = b)$. Optimal partitions with $R_{tat} = b$ require the partitions about twice as long as optimal partitions required with $R_{tat} = n$.

As a conclusion to the performed experiments, we suggest that the optimization of the feature processing chain parameters may provide a significant improvement of the classification quality. The number of the used features after optimization is much smaller then 33, as underscored in the last row of Table 2. It means that the feature processing, training and classification are done faster than if using all features.

Though the optimization runs require more computing time in comparison to direct processing, this process must be applied only once for each user category and can be done in the background. It has already been shown in [11] that not a single fixed set of features can be recommended for all classification tasks. So the feature processing chain must be rearranged for each music category for optimal classification. We can confirm this suggestion: No optimal partition size and no optimal tatum reduction method can be generally chosen for all possible music categories. Another point is that different user categories or other categorization tasks have different complexities. Even with the application of sophisticated optimization algorithms, the success rate has always some upper bound which cannot be exceeded. As the worst case, consider the user who creates the category ground truth in a completely random way.

6 Conclusion and Outlook

Music classification and navigation through large music collections is a complex problem which contains several steps - from the extraction of features to their processing, the classification training and validation. Each step is very important and a weak implementation of only one step leads to a significant reduction of the classification quality. In this paper, we started with a rather inclusive set of audio signal features, selected a classification method and attempted to optimize the feature processing configuration. The experiments have demonstrated that evolution strategies are a good choice to find those parameter settings, which are optimal for a given personal user category. Several tendencies of the parameter impacts were discovered.

For future research, we are going to construct a more complex feature processing chain, adding statistical dimensionality reduction methods such as principal components analysis, correlation analysis and feature extraction from different song segments. On the other side, the optimization algorithm itself can be examined more thoroughly. Self-adaptation and a crossover operator are possible concepts to investigate. Also experiments with other classification algorithms can be worthwhile. Finally not only standard measures like accuracy / precision can be used to evaluate the classifiers, but also runtime metrics and user-given feedback.

References

1. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, USA (1995)
2. Box, G., Jenkins, G.: Time Series Analysis: Forecasting and Control. Holden Day (1970)
3. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley-Interscience, Chichester (2000)
4. Flexer, A., Gouyon, F., Dixon, S., Widmer, G.: Probabilistic Combination of Features for Music Classification. In: Proc. of the 7th International Conference on Music Information Retrieval (ISMIR), pp. 111–114 (2006)
5. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.: Feature Extraction, Foundations and Applications. Springer, Heidelberg (2006)
6. Hall, M.: Correlation-based Feature Selection Machine Learning. PhD thesis, University of Waikato, New Zealand (1998)
7. McLachlan, G.: Discriminant Analysis and Statistical Pattern Recognition. Wiley Interscience, Chichester (1992)
8. Mierswa, I., Morik, K.: Automatic Feature Extraction for Classifying Audio Data. Machine Learning Journal 58, 127–149 (2005)
9. Ong, B.: Structural Analysis and Segmentation of Music Signals. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain (2006)
10. Peeters, G.: A Large Set of Audio Features for Sound Description (Similarity and Classification) in the CUIDADO Project. IRCAM, France (2004)
11. Pohle, T., Pampalk, E., Widmer, G.: Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories. In: Fourth International Workshop on Content-Based Multimedia Indexing (2005)
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
13. Rasmussen, C.: The Infinite Gaussian Mixture Model. In: Advances in Neural Information Processing Systems, pp. 554–560. MIT Press, Cambridge (2000)
14. Rudolph, G.: An Evolutionary Algorithm for Integer Programming. Parallel Problem Solving from Nature – PPSN III, 139–148 (1994)
15. Scaringella, N., Zoia, G., Mlynek, D.: Automatic Genre Classification of Music Content. IEEE Signal Processing Magazine 23, 133–141 (2006)
16. Seppänen, J., Eronen, A., Hiipakka, J.: Joint Beat and Tatum Tracking from Music Signals. In: Proc. of the 7th International Conference on Music Information Retrieval (ISMIR), Victoria, pp. 23–28 (2006)
17. Smith, L.: A Tutorial on Principal Components Analysis (2002)
18. Theimer, W., Vatolkin, I., Botteck, M., Buchmann, M.: Content-based Similarity Search and Visualization for Personal Music Categories. In: Sixth International Workshop on Content-Based Multimedia Indexing, London, pp. 9–16 (2008)
19. Theimer, W., Vatolkin, I., Eronen, A.: Definitions of Audio Features for Music Content Description. Algorithm Engineering Report TR08-2-001, Technische Universität Dortmund (2008)
20. Tzanetakis, G., Cook, P.: Musical Classification of Audio Signals. IEEE Transactions on Speech and Audio Processing 10, 293–302 (2002)
21. Vatolkin, I., Theimer, W.: Introduction to Methods for Music Classification Based on Audio Data, Technical Report NRC-TR-2007-012, Nokia Research Center (2007)

Author Index