Studies in Computational Intelligence 136

Carlos Cotta
Marc Sevaux
Kenneth Sörensen (Eds.)

Adaptive and Multilevel Metaheuristics

Springer

Studies in Computational Intelligence, Volume 136

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 117. Da Ruan, Frank Hardeman and Klaas van der Meer (Eds.)
Intelligent Decision and Policy Making Support Systems, 2008
ISBN 978-3-540-78306-0

Vol. 118. Tsau Young Lin, Ying Xie, Anita Wasilewska and Churn-Jung Liau (Eds.)
Data Mining: Foundations and Practice, 2008
ISBN 978-3-540-78487-6

Vol. 119. Slawomir Wiak, Andrzej Krawczyk and Ivo Dolezel (Eds.)
Intelligent Computer Techniques in Applied Electromagnetics, 2008
ISBN 978-3-540-78489-0

Vol. 120. George A. Tsihrintzis and Lakhmi C. Jain (Eds.)
Multimedia Interactive Services in Intelligent Environments, 2008
ISBN 978-3-540-78491-3

Vol. 121. Nadia Nedjah, Leandro dos Santos Coelho and Luiza de Macedo Mourelle (Eds.)
Quantum Inspired Intelligent Systems, 2008
ISBN 978-3-540-78531-6

Vol. 122. Tomasz G. Smolinski, Mariofanna G. Milanova and Aboul-Ella Hassanien (Eds.)
Applications of Computational Intelligence in Biology, 2008
ISBN 978-3-540-78533-0

Vol. 123. Shuichi Iwata, Yukio Ohsawa, Shusaku Tsumoto, Ning Zhong, Yong Shi and Lorenzo Magnani (Eds.)
Communications and Discoveries from Multidisciplinary Data, 2008
ISBN 978-3-540-78732-7

Vol. 124. Ricardo Zavala Yoe
Modelling and Control of Dynamical Systems: Numerical Implementation in a Behavioral Framework, 2008
ISBN 978-3-540-78734-1

Vol. 125. Larry Bull, Bernadó-Mansilla Ester and John Holmes (Eds.)
Learning Classifier Systems in Data Mining, 2008
ISBN 978-3-540-78978-9

Vol. 126. Oleg Okun and Giorgio Valentini (Eds.)
Supervised and Unsupervised Ensemble Methods and their Applications, 2008
ISBN 978-3-540-78980-2

Vol. 127. Régie Gras, Einoshin Suzuki, Fabrice Guillet and Filippo Spagnolo (Eds.)
Statistical Implicative Analysis, 2008
ISBN 978-3-540-78982-6

Vol. 128. Fatos Xhafa and Ajith Abraham (Eds.)
Metaheuristics for Scheduling in Industrial and Manufacturing Applications, 2008
ISBN 978-3-540-78984-0

Vol. 129. Natalio Krasnogor, Giuseppe Nicosia, Mario Pavone and David Pelta (Eds.)
Nature Inspired Cooperative Strategies for Optimization (NICSO 2007), 2008
ISBN 978-3-540-78986-4

Vol. 130. Richi Nayak, Nikhil Ichalkaranje and Lakhmi C. Jain (Eds.)
Evolution of the Web in Artificial Intelligence Environments, 2008
ISBN 978-3-540-79139-3

Vol. 131. Roger Lee and Haeng-Kon Kim (Eds.)
Computer and Information Science, 2008
ISBN 978-3-540-79186-7

Vol. 132. Danil Prokhorov (Ed.)
Computational Intelligence in Automotive Applications, 2008
ISBN 978-3-540-79256-7

Vol. 133. Manuel Graña and Richard J. Duro (Eds.)
Computational Intelligence for Remote Sensing, 2008
ISBN 978-3-540-79352-6

Vol. 134. Ngoc Thanh Nguyen and Radoslaw Katarzyniak (Eds.)
New Challenges in Applied Intelligence Technologies, 2008
ISBN 978-3-540-79354-0

Vol. 135. Hsinchun Chen and Christopher C. Yang (Eds.)
Intelligence and Security Informatics, 2008
ISBN 978-3-540-69207-2

Vol. 136. Carlos Cotta, Marc Sevaux and Kenneth Sörensen (Eds.)
Adaptive and Multilevel Metaheuristics, 2008
ISBN 978-3-540-79437-0

Carlos Cotta
Marc Sevaux
Kenneth Sörensen
(Eds.)

Adaptive and Multilevel Metaheuristics

Springer

Carlos Cotta
ETSI Informatica (3.2.49)
Campus de Teatinos
Universidad de Malaga
29071, Malaga
Spain
E-mail: ccottap@lcc.uma.es

Marc Sevaux
University of South-Brittany
CNRS, FRE 2734, LESTER
Centre de Recherche - BP 92116
F-56321 Lorient cedex
France
E-mail: marc.sevaux@univ-ubs.fr

Kenneth Sörensen
Fellow of the Flemish Fund for Scientific Research
Centre for Industrial Management
Katholieke Universiteit Leuven
Celestijnenlaan 300a
3001 Leuven
Belgium
E-mail: kenneth.sorensen@cib.kuleuven.be

ISBN 978-3-540-79437-0 e-ISBN 978-3-540-79438-7

DOI 10.1007/978-3-540-79438-7

Studies in Computational Intelligence ISSN 1860949X

Library of Congress Control Number: 2008925773

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Adapt or Perish, now as ever, is nature's inexorable imperative
— H.G. Wells (1866-1946)

Preface

The last decades have witnessed a profound change in search and optimization technologies. In those problem domains where complexity results deemed exact techniques unaffordable, the use of metaheuristics has steadily gained popularity and usage. Nowadays, these techniques exhibit a remarkable success record, and are considered cutting-edge methods for solving hard optimization problems. Thus, whenever new problem domains arise metaheuristics are one of the primary weapons in our solving arsenal.

One of the keystones in practical metaheuristic problem-solving is the fact –repeatedly shown in both theory and practice– that tuning the optimization technique to the problem under consideration is crucial for achieving top performance. This tuning/customization is usually on the hands of the algorithm designer, and despite some methodological attempts, it largely remains an scientific art. Needless to say, there exist a number of very useful guidelines available in the literature for algorithmic parameterization, operator design, etc, but these guidelines are in general heuristic in nature.

A longly pursued goal in the field of metaheuristics has been transferring a part of this customization effort to the algorithm itself, endowing it with smart mechanisms for self-adapting to the problem. These mechanisms can involve different aspects of the algorithm, such as for example, self-adjusting the parameters, self-adapting the functioning of internal components, evolving search strategies, etc. While some theoretical results set out limitations on the general robustness of such mechanisms, their usefulness in specific problem classes has been verified. This volume presents recent advances in the area of self-adaptation in metaheuristic optimization. Most articles in this collection arose from a dedicated workshop held in Málaga, Spain, in November 2006 under the auspices of the European Chapter on Metaheuristics (EU/ME).

The volume is organized in two blocks. The first one comprises two review articles that survey the major aspects in the area of self-adaptive metaheuristics, namely hyperheuristics and self-adaptation in evolutionary heuristics. The first paper is authored by K. Chakhlevitch and P. Cowling, and overviews hyperheuristics, a multi-level metaheuristic approach in which an upper heuristic

layer controls the application of some underlying heuristics depending upon the characteristics of the region of the solution space currently under exploration. Different strategies for designing hyperheuristics are discussed, providing pointers to applications along the way. The second paper is authored by J.E. Smith, and focuses on how self-adaptation mechanisms in evolutionary algorithms may be used to control the parameters defining crossover and mutation, as well as the very definition of local search operators used within hybrid evolutionary algorithms.

The second block contains novel techniques involving self-adaptation, or novel applications of these. Araya, Neveu and Riff consider hyperheuristics for strip packing problems. Their hyperheuristic performs hill-climbing on a sequence of greedy low-level heuristics for the mentioned problem, obtaining results that often outperform other heuristics for the strip packing problem. Boutillon, Roland and Sevaux consider a simulated annealing approach to a hardware problem (the optimization of a finite impulse response filter). They focus on the acceptance schedule of the algorithm, and propose a probability-driven schedule that is shown to be competitive with temperature-driven strategies. Brunato and Battiti present an adaptive random search scheme termed reactive affine shaker that adapts the search region via affine transformations. The modifications are done on the basis of information gathered from trial points, and as shown by the experimental results they result in a promising approach for continuous optimization.

Geiger and Wenger consider an adaptive multi-objective vehicle routing problem in which different interacting vehicle agent place bids for orders that are offered in a marketplace. A decider agent communicates with a human user to build a preference model of her preferences. Landa-Silva and Le also consider a multi-objective problem, in this case in the area of nurse scheduling. Their approach uses a decoder mechanisms that incorporates a self-mutation mutation operator for repairing hard-constraint violations. Olague, Dunn and Lutton consider a problem in the area of computer vision, and approach it with an adaptive strategy based on the evolution of interacting co-adapted subcomponents for the problem at hand. They consider issues such as the computation and redistribution of fitness values, the preservation of diversity, and the aggregation of individuals to form composite solutions.

Santana, Larrañaga and Lozano study the issue of adaptation within the framework of estimation of distribution algorithms (EDAs). They consider a framework in which the underlying probabilistic model can change during evolution, and show how such a dynamic EDA can outperform a static EDA on the satisfiability problem. Sierra Urrechu and Santibáñez Koref also consider an EDA approach, applied in this case to search directions within the search space rather than to solutions as usual. This approach can effectively self-adapt the trajectory of a local-searcher on continuous optimization problems. Cooren, Clerk and Siarry propose a parameter-free particle swarm algorithm that adapts its parameters according to information collected during the optimization process.

It is shown that this approach is competitive with respect to other swarm-based and evolution-based methods.

Sörensen, Sevaux and Schittekat contribute a position paper in which they examine commercial vehicle routing packages and identify multiple-neighborhood search as the key feature for adaptation. They present the thesis that this convergent feature is essential to the flexibility and adaptability of these approaches to different problems, and outline the need for adaptive hyperheuristics to supplement –and eventually, substitute– the role of the human user as designer. Finally, Souffriau, Vansteenwegen, Vanden Berghe and Van Oudheusden consider a hyperheuristic approach based on ant-colony optimization (ACO) with application to a routing problem. A genetic algorithm tunes the parameters of the ACO algorithm on a training set of problems, and the so-adapted algorithm is then tested on a different set of problems providing results very close to optimality.

Overall, this volume is intended both as a reference work for novel researchers in the area of self-adaptation in metaheuristics, and as an inspiring collection of state-of-the-art articles for researchers actively working in the field. We would like to thank all the people who made this volume possible, starting by the authors who contributed the technical content of the book. We also thank Dr. Antonio J. Fernández and Dr. José E. Gallardo from the University of Málaga for their invaluable participation in the organization of the seminal workshop held on this topic in Málaga on 2006. The financial help of the University of Málaga and the European Chapter on Metaheuristics is acknowledged too. Last but not least, thanks are due to Prof. Janusz Kacprzyk for his support to the development of this project, and to Dr. Thomas Ditzinger and the editorial staff of Springer for their kind attention and help.

Málaga (Spain), Lorient (France), Leuven (Belgium) Carlos Cotta
February 2008 Marc Sevaux
Kenneth Sörensen

Contents

List of Contributors

Ignacio Araya
Project COPRIN, INRIA, Sophia-Antipolis, France
ignacio.araya@sophia.inria.fr

Roberto Battiti
Dipartimento di Ingegneria e Scienza dell'Informazione, Università di Trento, via Sommarive 14, I-38100 Trento, Italy
battiti@dit.unitn.it

Emmanuel Boutillon
Université Européenne de Bretagne, UBS - Lab-STICC - Centre de Recherche, F-56321 Lorient France
emmanuel.boutillon@univ-ubs.fr

Mauro Brunato
Dipartimento di Ingegneria e Scienza dell'Informazione, Università di Trento, via Sommarive 14, I-38100 Trento, Italy
brunato@dit.unitn.it

Konstantin Chakhlevitch
CASS Business School, City University, London EC1Y 8TZ, UK
konstantin.chakhlevitch.1@city.ac.uk

Maurice Clerc
LiSSi, E.A. 3956 Université de Paris XII, 61 avenue du Général de Gaulle, 94010 Créteil, France
maurice.clerc@writeme.com

Yann Cooren
LiSSi, E.A. 3956 Université de Paris XII, 61 avenue du Général de Gaulle, 94010 Créteil, France
cooren@univ-paris12.fr

Peter Cowling
Department of Computing, University of Bradford, Bradford BD7 1DP, UK
P.I.Cowling@Bradford.ac

Enrique Dunn
CICESE, Km. 107 carretera Tij-Eda, 22860 Ensenada, México
edunn@cicese.mx

Martin Josef Geiger
Lehrstuhl für Industriebetriebslehre, Universität Hohenheim, 70593 Stuttgart, Germany
mjgeiger@uni-hohenheim.de

Dario Landa-silva
School of Computer Science, The University of Nottingham, UK
jds@cs.nott.ac.uk

Pedro Larrañaga
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Paseo Manuel de Lardizabal 1, 20080 Donostia - San Sebastian, Spain
pedro.larranaga@ehu.es

Khoi N. Le
School of Computer Science, The University of Nottingham, UK
kxl@cs.nott.ac.uk

Jose A. Lozano
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Paseo Manuel de Lardizabal 1, 20080 Donostia - San Sebastian, Spain
ja.lozano@ehu.es

Evelyne Lutton
INRIA Rocquencourt, Le Chesnay Cedex, France
evelyne.lutton@inria.fr

Bertrand Neveu
Project COPRIN, INRIA, Sophia-Antipolis, France
bertrand.neveu@sophia.inria.fr

Gustavo Olague
CICESE, Km. 107 carretera Tij-Eda, 22860 Ensenada, México
olague@cicese.mx

María-Cristina Riff
Department of Computer Science, Universidad Técnica Federico Santa María, Valparaíso, Chile
maria-cristina.riff@inf.utfsm.cl

Christian Roland
Université Européenne de Bretagne, UBS - Lab-STICC - Centre de Recherche, F-56321 Lorient, France
christian.roland@univ-ubs.fr

Roberto Santana
Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Paseo Manuel de Lardizabal 1, 20080 Donostia - San Sebastian, Spain
rsantana@si.ehu.es

Iván Santibáñez Koref
Bionics and Evolutionary Techniques Dept., Technische Universität Berlin, D-13355 Berlin, Germany
isk@bionik.tu-berlin.de

Patrick Schittekat
University of Antwerp, Faculty of Applied Economics, Prinsstraat 13, 2000 Antwerp, Belgium
patrick.schittekat@ua.ac.be

Marc Sevaux
Université Européenne de Bretagne, UBS - Lab-STICC - Centre de Recherche, F-56321 Lorient, France
marc.sevaux@univ-ubs.fr

Patrick Siarry
Laboratoire Images, Signaux et Systèmes Intelligents, LiSSi, E.A. 3956 Université de Paris XII, 61 avenue du Général de Gaulle, 94010 Créteil, France
siarry@univ-paris12.fr

James E. Smith
School of Computer Science, University of the West of England at Bristol, UK
james.smith@uwe.ac.uk

Kenneth Sörensen
Katholieke Universiteit Leuven, Centre for Industrial Management, Celestijnenlaan 300A, 3001 Leuven (Heverlee), Belgium
kenneth.sorensen@cib.kuleuven.be

Wouter Souffriau
KaHo Sint–Lieven,
Information Technology,
Gebroeders Desmetstraat 1, 9000
Gent, Belgium
wouter.souffriau@kahosl.be

Alejandro Sierra Urrecho
Department of Computer Engineering,
Universidad Autónoma de Madrid,
28049 Madrid, Spain
alejandro.sierra@uam.es

Greet Vanden Berghe
KaHo Sint–Lieven, Information
Technology, Gebroeders
Desmetstraat
1, 9000 Gent, Belgium
greet.vandenberghe@kahosl.be

Pieter Vansteenwegen
Katholieke Universiteit Leuven,
Centre for Industrial Management,
Celestijnenlaan 300A,
3001 Leuven (Heverlee), Belgium
pieter.vansteenwegen@cib.
kuleuven.be

Dirk Van Oudheusden
Katholieke Universiteit Leuven,
Centre for Industrial Management,
Celestijnenlaan 300A,
3001 Leuven (Heverlee), Belgium
dirk.vanoudheusden@cib.
kuleuven.be

Wolf Wenger
Lehrstuhl für Industriebetriebslehre,
Universität Hohenheim,
70593 Stuttgart, Germany
w-wenger@uni-hohenheim.de

Part I
Reviews of the Field

Hyperheuristics: Recent Developments

Konstantin Chakhlevitch[1] and Peter Cowling[2]

[1] CASS Business School, City University, London EC1Y 8TZ, UK
Konstantin.Chakhlevitch.1@city.ac.uk
[2] Department of Computing, University of Bradford, Bradford BD7 1DP, UK
P.I.Cowling@Bradford.ac.uk

Keywords: Hyperheuristics, multilevel heuristics, greedy heuristics, learning.

1 Introduction

Given their economic importance, there is continuing research interest in providing better and better solutions to real-world scheduling problems. The models for such problems are increasingly complex and exhaustive search for optimal solutions is usually impractical. Moreover, difficulty in accurately modelling the problems means that mathematically "optimal" solutions may not actually be the best possible solutions in practice. Therefore heuristic methods are often used, which do not guarantee optimal or even near optimal solutions. The main goal of heuristics is to produce solutions of acceptable quality in reasonable time. The problem owners often prefer simple, easy to implement heuristic approaches which do not require significant amount of resources for their development and implementation [12]. However, such individual heuristics do not always perform well for the variety of problem instances which may be encountered in practice. There is a wide range of modern heuristics known from the literature which are specifically designed and tuned to solve certain classes of optimisation problems. These methods are based on the partial search of the solution space and often referred as *metaheuristics*.

Although tailor-made metaheuristic algorithms have proved to be very effective for solving various combinatorial optimisation problems, their application is usually limited to particular problem domains. Metaheuristics incorporate information specific for the problem and "require extensive knowledge in both the problem domain and appropriate heuristic techniques" [21]. Therefore such methods are often quite expensive to implement. Metaheuristic approaches that perform well on a particular real-world problem, may not work at all or may produce very poor solutions for other problems or even for other instances of the same problem. Such limitations can become especially critical in situations when problem data and business requirements change frequently over time. This can make a metaheuristic even more expensive because it should be properly maintained.

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 3–29, 2008.
springerlink.com

Burke et al. in [12] note that many businesses are interested in "good enough-soon enough-cheap enough" solutions to their problems provided by easy-to-use and robust heuristic approaches rather than optimal or near optimal solutions achieved at the expense of the development of customised problem-specific methods such as metaheuristics and possibly using greatly simplified models. This is a primary motivation for development of generalised, domain-independent heuristic search techniques which have recently become known as *hyperheuristics* and have received an increased attention in the research community. The purpose of hyperheuristics is not to compete with state-of-the-art problem-specific approaches, but to provide a general framework able to deliver solutions of a good quality for a wide range of optimisation problems.

Another motivation for development of hyperheuristics comes from the fact that performance of different heuristics may vary significantly depending on the specific characteristics of the problem and problem instance under consideration. Moreover, individual heuristics may be particularly effective at certain stages of the solution process (i.e. when certain areas of the solution space are being explored) while performing poorly at any other stages. Therefore, it is fair to expect that several heuristics combined in a proper way may produce better solutions then if they are applied separately. A hyperheuristic can be defined as a *heuristic which chooses heuristics* [61]. In other words, a hyperheuristic operates in a *space of heuristics* choosing and applying one low level heuristic from a given *set* at each decision point. This is where the fundamental difference between hyperheuristics and metaheuristics lies since a metaheuristic is a heuristic which controls the search in a *space of solutions* performed by a single low level heuristic..

The term "hyperheuristic" was first introduced by Cowling et al. in [21]. They defined a hyperheuristic as an "approach that operates at a higher level of abstraction than metaheuristics and manages the choice of which low-level heuristic method should be applied at any given time, depending upon the characteristics of the region of the solution space currently under exploration". This means that the hyperheuristic itself does not search for a better solution to the problem. Instead, it selects at each step of the solution process the most promising simple low-level heuristic (or combination of heuristics) which is potentially able to improve the solution. On the other hand, if there is no improvement, i.e., a locally optimal solution is found, the hyperheuristic diversifies the search to another area of the solution space by selecting appropriate heuristics from the given set. Low-level heuristics usually represent the simple local search neighbourhoods or the rules used by human experts for constructing solutions. However, it is also possible that more complex heuristics such as metaheuristics can be considered at a lower level. All domain-specific information is concentrated in the set of low-level heuristics and the objective function. Hyperheuristics do not require knowledge of how each low-level heuristic works or the contents of the objective function of the problem (other than the value returned). It only needs to know the direction of the optimisation process (maximising or minimising) and analyses the value of one or more objective functions and, sometimes, the amount of

CPU time required to perturb the solution, which are returned by the low-level heuristic after its call.

Hyperheuristics have received much attention over the last 5 years or so and will likely remain a hot research topic for the near future. The first paper on hyperheuristics was presented by Fisher and Thompson [30] in 1961, but there was no other work in this area published between then and the 1990's, when a few related approaches were developed. The latter methods were mainly based on genetic algorithms which used indirect chromosome representation so that the chromosome encoded the method to solve a problem instead of the solution of a problem. However, the motivation for such methods was rather to overcome the difficulties related to solution encoding and maintaining the solutions feasibility than to develop general solution techniques able to tackle different optimisation problems. Nevertheless, the ideas used in these approaches created the basis for recent developments in hyperheuristics. We shall discuss them later in this chapter.

Based on the the original definition introduced by Cowling et al. [21], we shall use the following criteria to define a hyperheuristic:

1. A *hyperheuristic* is a *higher level heuristic* which manages a *set* of *low level heuristics*, of cardinality greater than one.
2. A hyperheuristic searches for a good *method* to solve the problem rather than for a good solution.
3. A hyperheuristic uses only *limited* problem-specific information (ideally this information includes only the number of low level heuristics for the problem and objective function(s) to be maximised or minimised).

The third criterion is the most crucial one since it defines the level of generality of the hyperheuristic approach as well as its potential robustness across different problem domains. Many techniques considered in this chapter match to the first two statements and fail to comply to the third one.

From our point of view, hyperheuristic approaches developed so far can be classified into the following categories: hyperheuristics based on the random choice of low level heuristics (Section 2), greedy and peckish hyperheuristics (Section 3), metaheuristic-based hyperheuristics (Section 4) and hyperheuristics employing learning mechanisms to manage low level heuristics (Section 5). We will also consider other generic solving methods which are closely related to hyperheuristics (Section 6).

2 Hyperheuristics Based on Random Selection

Hyperheuristics based on the random choice of low level heuristics from a given set have been widely represented in the literature. Table 1 provides a list of the approaches which fall in this class.

The "random" hyperheuristic is the oldest, the simplest, and easiest to implement of the family of hyperheuristics. It randomly selects one low level heuristic from a given set at each decision point. The selected low level heuristic is always

Table 1. Hyperheuristic methods based on random selection of low level heuristics

Approach	Papers	Details
Pure random	Cowling et al. [21], [23], [24] Kendall and Mohamad [46] Bai and Kendall [4] Burke et al. [9] Cowling and Chakhlevitch [18] Storer et al. [62]	Uniform selection of LLH*; either all LLH or only improving LLH are accepted
Random descent	Cowling et al. [21], [22] Soubeiga [61]	Improving LLH is applied repeatedly until it does not improve the solution
Unbiased random process	Fisher and Thompson [30], [31]	Multistart process where probabilities of LLH selection are adjusted after each run
Monte Carlo	Ayob and Kendall [3] Bai and Kendall [4] Soubeiga [61] Chakhlevitch [16]	Probability of accepting LLH is a function of the difference between old and new objective values; includes simulated annealing
Random with deterministic acceptance	Kendall and Mohamad [45], [46]	Acceptance rules are based on the distance between two solutions; includes Great Deluge and record-to record-travel methods

*LLH stands for low level heuristic(s).

applied, even if it does not produce any improvement or worsens the current solution of the problem.

Pure random hyperheuristics have been tested by many researchers for different optimisation problems including the sales summit scheduling problem [21], the project presentation scheduling problem [23], the nurse rostering problem [24], the channel assignment problem in mobile communications [46], the shelf space allocation problem [4], university course timetabling problem [9], the trainer scheduling problem [18] and job shop scheduling problem [62]. Note that pure random approach is usually used as a point of comparison for the performance of individual low level heuristics (where it is usually better) or other, more intelligent hyperheuristic methods (where it is usually worse).

The main disadvantage of a purely random approach is that the quality of the solution obtained depends on the chance of selecting a "good" sequence of low level heuristics. In order to avoid the search becoming trapped into poor regions of the solution space, modifications to the pure random approach are needed. These modifications may concern the rule of accepting non-improving low level heuristics during the hyperheuristic run or the way of applying low level heuristic at each decision point. In [21] and [22], Cowling et al. compare different versions of random hyperheuristics applied to a practical problem of scheduling meetings for a sales summit. One version of hyperheuristic applies a randomly selected low level heuristic only once at each iteration (simple random hyperheuristic), another one conducts simple local search by applying a

randomly selected low level heuristic in a descent fashion, i.e. the low level heuristic is reapplied as long as it continues to produce an improvement to the current solution (descent hyperheuristic). The results of experiments show that descent hyperheuristics perform better than simple versions. In addition, two acceptance rules are considered: accept all low level heuristics or only improving ones. Note that the "only improving" strategy may cause the hyperheuristic to get stuck in a local optimum at the early stages of the search when there is no a single low level heuristic in the set which is able to improve the current solution. This situation is likely to happen when the number of low level heuristics used is small. The natural way to overcome the problems related to both extreme acceptance strategies is to allow non-improving low level heuristics to be applied with some probability.

In their seminal work, Fisher and Thompson [30], [31] use the term "unbiased random process" for their random hyperheuristic. They consider a classical job shop scheduling problem [7] and just two simple priority rules to select the next job to be scheduled at each machine. In [31], the sequences of rules constructed by the unbiased random process, produced better schedules than both rules in the set applied separately. Fisher and Thompson [31] use reinforcement learning techniques [63] in order to determine the probabilities of selecting specific decision rules at any point of the scheduling process. Each time a new sequence of rules is generated, the resulting schedule produced by this sequence is compared to the standard schedule (which is usually the best schedule found so far). If the current schedule is better than the standard, it becomes a new standard, otherwise the standard schedule remains the same. Then the points of difference in the sequences of rules for both schedules are determined and the probabilities of selecting rules at these points are adjusted and used for the next schedule generation. The results reported in [31] show an improved average performance of the method with learning if compared to a purely random approach. Fisher and Thompson conclude that probabilistic learning approach might be more effective if a larger number of decision rules is combined rather than just two rules considered in [31].

A number of hyperheuristics recently presented in the literature are based on Monte Carlo methods. A general Monte Carlo method [33] uses probability for accepting a new non-improving solution which decreases when the difference δ in objective values between the new and the current (best) solutions increases (for minimisation problem). In the context of a hyperheuristic, the probability of accepting non-improving low level heuristic is considered. This probability can be computed in different ways. Ayob and Kendall in [3] apply a Monte Carlo hyperheuristic to optimise electrical component placement on a printed circuit board. The set of 6 low level heuristics represents different versions of simple swap moves. The authors consider linear and exponential functions of δ to define acceptance probabilities for non-improving low level heuristics. The best results are achieved for an exponential function where time factors are taken into account for calculating probabilities, i.e. an acceptance probability at any decision point depends not only on δ, but on the time elapsed since a start of

a hyperheuristic and on the number of consecutive non-improving iterations. Such a method for calculating probabilities is similar to that used in another approach from the Monte Carlo family, simulated annealing [1], where the acceptance probability is a function of δ and control parameter (temperature) which gradually decreases as the number of iterations grows. Bai and Kendall in [4] develop a simulated annealing based hyperheuristic to solve a shelf space allocation problem. Combining 12 low level heuristics, their hyperheuristic approach outperforms two versions of simulated annealing metaheuristic and produce high quality results for different problem instances. Strong performance of simulated annealing based hyperheuristics for a sales summit scheduling problem and for a trainer scheduling problem are reported by Soubeiga in [61] and Chakhlevitch in [16], respectively.

Another example of random hyperheuristics are hyperheuristics based on the variants of Great Deluge algorithm [27], considered by Kendall and Mohamad in [45] and [46]. A Great Deluge hyperheuristic randomly selects a low level heuristic at each iteration and applies it if the objective value returned by the low level heuristic is better than some specified threshold level. The level is initially set to the objective value of the starting solution and then adjusted after each iteration, i.e. it decreases (for minimisation problems) at a fixed rate [45]. This strategy allows non-improving moves to be accepted frequently at the early stages of a hyperheuristic run and very occasionally towards the end. In [46], another method to control accepting non-improving low level heuristics is used: a low level heuristic is accepted only if its returned objective value is reasonably close to the objective value of the current solution. This is implemented by introducing a parameter which represents the maximum possible distance between two solutions. Both hyperheuristics produce results of a good quality for the channel assignment problem (see [45] and [46]).

Random hyperheuristics are simple and fast, and can be easily implemented and applied to any optimisation problem. The results achieved by basic random hyperheuristics can be used as benchmarks for evaluating other hyperheuristic approaches. Hybridisation with more advanced techniques for accepting low level heuristics make random hyperheuristics competitive with other approaches. This is probably the case since the outcome of applying each low level heuristic depends on effectively random factors and their behaviour at different decision points is difficult to predict.

3 Greedy and Peckish Hyperheuristics

A basic greedy hyperheuristic simply selects and applies at each decision point the low level heuristic which produces the largest improvement to the current objective value (or the smallest deterioration if no improving low level heuristic exists at some iteration). Two versions of the greedy strategy can be used: one accepts only improving low level heuristics and another allows non-improving low level heuristics to be applied. The second version is advantageous since it prevents a hyperheuristic from stopping too early when no improving low

level heuristic is available in the set. Note that a greedy hyperheuristic requires preliminary evaluation of each low level heuristic in the set in order to select the best one which makes it much slower than a hyperheuristic based on the random choice. Greedy hyperheuristics are considered by Cowling et al. [21] – [24] and by Cowling and Chakhlevitch [18], [19] and their results are mainly used as benchmarks for other methods.

The main disadvantage of a greedy hyperheuristic is its limited ability to effectively explore the search space leaving many regions with potentially strong solutions unvisited. To overcome problems associated with local optima, Cowling and Chakhlevitch [18], [19] develop a group of "peckish" hyperheuristics which combine greedy and random mechanisms for managing the choice of low level heuristics. A peckish hyperheuristic randomly selects a low level heuristic from the candidate list of the "best" (not necessarily improving) ones. The length of the candidate list may be adjusted in order to achieve a good ratio between intensification and diversification in the search. The authors consider four versions of peckish hyperheuristics using both static and dynamic candidate lists and apply them to the trainer scheduling problem

Note that peckish hyperheuristics may be particularly useful when a large set of low level heuristics is used since they present a scalable method capable of handling any number of low level heuristics. In [18] and [19], the authors present a generic idea for constructing a large set of low level heuristics for complex little-studied optimisation problems. For such problems, there is no obvious choice of low level heuristics and traditional neighbourhoods (swap, insert or replace moves) are not easily applicable. Tackling the real-world trainer scheduling problem, Cowling and Chakhlevitch show how a set of low level heuristics can be formed by combining simple selection rules for events and resources. See also [16] for more details.

Greedy and peckish hyperheuristics can be readily applied for different optimisation problems due to their simplicity and high level of generality. However, their slow speed makes them unfavourable for the problems where the time to construct solutions is a crucial factor.

4 Metaheuristic-Based Hyperheuristics

A conventional metaheuristic is a local search based method which operates in a *solution* space of the problem and employs some strategy to escape local optima. Taking into account a proven record of successful applications of metaheuristics to solving complex real-world optimisation problems, the question of how effectively metaheuristics can perform the search over a *heuristic* space is of a great research interest. Various metaheuristic approaches and their hybrids have been tested as a high level heuristic selectors in the last few years and we refer to them as metaheuristic-based hyperheuristics in this section. We start with hyperheuristics based on genetic algorithms (GAs) which have created the foundation for current research in hyperheuristics.

Table 2. Hyperheuristics based on genetic algorithms

Papers	Details
Fang et al. [28] Hart et al. [42], [43]	Indirect GA with a block structure of the chromosome; each block contains combination of LLH and domain-specific information
Norenkov and Goodman [55] Dorndorf and Pesch [25] Hart and Ross [41]	The length (or dimensions, in case of matrix representation) of a chromosome is determined by the value(s) of problem-specific parameter(s)
Terashima-Marín et al. [64] Hart and Ross [41]	LLHs performing different actions are combined in the chromosome
Cowling et al. [20] Han et al. [39] Han and Kendall [37], [38]	A chromosome determines a sequence of LLHs and the order of their application; the length of the chromosome is either fixed or adaptively adjusted
Ross et al. [59]	A chromosome encodes characteristics of the problem instances together with associated LLHs

4.1 GA-Based Hyperheuristics

Early efforts to search for an effective solution method for a problem rather than for a good solution are related to the development of GAs with indirect chromosome encoding. A chromosome in a traditional GA encodes a solution to the problem directly (by means of binary strings, permutations, etc.). However, the solutions to many real-world problems have a very complex structure which makes the direct encoding extremely difficult. The other disadvantages of direct encoding are a large length of the chromosomes for large problems and the need of specific repair operators to maintain the feasibility of solutions. A number of indirect GAs developed in 1990s were aimed to overcome these limitations.

In indirect GAs each chromosome represents the way a solution is constructed rather than the solution itself. In [64], the chromosome represents a sequence of heuristics to be applied to the initial solution. The ith gene of the chromosome encodes the heuristic number in the set of possible heuristics and indicates that this heuristic will be applied at the ith step of generating a new solution. Note that a chromosome in an indirect GA may also encode the order in which the (single) heuristic is applied, but we do not consider such an approach in this review. Table 2 provides a summary of the methods which can be classified as GA-based hyperheuristics.

The idea of indirect encoding was first implemented by Norenkov [54] for a scheduling problem connected with the CAD system hardware design and by Fang et al. [28] for an open shop scheduling problem. Fang et al. [28] use a set of eight simple dispatching rules as low level heuristics. The chromosome is organized as a sequence of pairs of genes. The first gene in each pair represents the heuristic and the second one represents the uncompleted job whose operation will be scheduled by applying this heuristic. The results produced by the GA

are very close to the best previously found for the most of the benchmark open shop problems and even better for some instances.

Another successful application of indirect GA is reported by Hart et al. [42], [43]. They present a GA-based approach to tackle the real-world scheduling problem of a chicken processing company. The problem is heavily constrained and required several days of work of a human expert to produce a practical schedule. The goal is to schedule chicken catching squads and lorries to deliver a set of orders to the factories to ensure that the factories will be supplied with the birds continuously throughout the day. The problem is decomposed into two stages and two separate GAs are implemented in each stage respectively. We mention here only the first one, an *assignment GA*, which performs the assignment of tasks to squads. An assignment GA uses an indirect chromosome representation thus evolving an assignment strategy and then applying that strategy to construct a schedule. The strategy incorporates the combinations of two heuristics for each order: one heuristic for splitting the order into tasks and the second heuristic for assigning the tasks to squads. The chromosome consists of four sections. The first section contains problem specific information, expressing certain fixed criteria which must be satisfied by every solution. Including such information into the chromosome allows the reduction of the search space. The second section contains the permutation of the factory orders, i.e. the sequence in which the orders will be processed by the strategy. The third and the fourth sections of the chromosome specify for each order the splitting and the assignment heuristics respectively. The GA uses specially designed crossover and mutation operators for each section of the chromosome. An assignment GA in [42], [43] produces practical schedules with almost all constraints satisfied.

Norenkov and Goodman in [55] further develop the approach introduced in [54] and refer to it as a Heuristic Combination Method (HCM). In [55], HCM is applied to solve multistage job-shop scheduling problems. The authors consider two parts of a process of schedule synthesis, specifically job ordering and the assignment of the jobs to servers, and define a set of simple heuristic rules for each part. The composition of the rules for both parts forms the set of heuristics, and the objective is to find the optimal sequence of application of these heuristics. The chromosome is represented as a matrix of size $q * N$,where N is the number of jobs and q is the number of successive service stages each job passes during its processing. The schedule is constructed consecutively for each service stage by adding one job in each step. The element (i, j) of the matrix refers to the heuristic which is applied on the jth step of schedule synthesis at the ith stage of service. The jth step here means that $j - 1$ jobs have been already scheduled and job j is due to be placed into the schedule. Given such a representation, the authors define specific horizontal and vertical crossover operators (crossover applied to rows and columns of the matrices respectively). They present several evolutionary algorithms based on the above chromosome representation which have been successfully applied to some benchmark job-shop scheduling problems.

A GA developed by Dorndorf and Pesch [25] evolves the sequence of low level heuristics for minimising makespan in job shop scheduling problems. The set

of low level heuristics consists of 12 well-known priority rules. A chromosome consists of $n-1$ genes, where n is the number of operations to be scheduled, and each gene encodes the priority rule to be applied to schedule one operation. The crossover operator exchanges substrings of priority rules in two chromosomes and the mutation operator replaces the rule in the randomly selected position of the chromosome with another, randomly selected rule. Although the indirect GA loses to other heuristic methods (such as the Shifting Bottleneck heuristic [2], a conventional GA and simulated annealing) both in solution quality and in speed, it is easy to implement and it shows robustness to problem changes. Hart and Ross [41] extend the approach presented in [25] and develop the method they call HGA (heuristically-guided GA) to solve a dynamic job shop problem. Each gene of a chromosome in HGA now encodes a pair (*Method, Heuristic*) where *Method* represents one of the two algorithms used to calculate the conflicting set of operations at each iteration (Dorndorf and Pesch in [25] consider only one such algorithm) and *Heuristic* is one of the 12 priority rules used to select an operation from the conflicting set. Hart and Ross show that switching between two scheduling methods during schedule construction is beneficial and their method outperforms other heuristics for many instances.

Terashima-Marín et al. [64] investigate the effectiveness of an indirect GA applied to a real-world examination timetabling problem. The authors consider the performance of Brelaz's well-known graph-colouring algorithm [6] combined with heuristics for handling different types of problem constraints. The performance varies for the different problems from the test set and depends on the heuristics chosen. Since there is no evidence which combination of heuristics will be the most suitable for solving any particular problem, Terashima-Marín et al. develop an indirect GA to evolve combinations of heuristics and find the best one for any instance. They specify three different algorithms for solving graph colouring problems and two sets of heuristics for decision making. At the first decision point the nodes of the graph are ordered for the colouring algorithm (variable ordering) and at the second decision point, the order algorithm will select the colours for a node (value ordering). The chromosome encoding is the 10-position array of characters which represent two combinations of graph colouring method with variable ordering and value ordering heuristics, the condition for switching from the first combination to the second, the parameter for the specific condition, and the indicator of the method of handling the constraints. The graph colouring methods include Brelaz's algorithm [6] and two procedures which involve backtracking and forward checking mechanisms respectively. The purpose of the two combinations of the methods and the heuristics in the chromosome is to handle the situations when the first combination becomes inefficient (performs too many backtracking steps, exceeds time limit) or just to mix two combinations in the hope of obtaining a better solution. The solutions obtained for all tested problems by applying GA with such chromosome representation are superior to those produced by Brelaz's graph colouring algorithm (see [64]).

The GA-based approaches considered so far are rather domain-specific. Indirect GAs are designed to solve different instances of the particular problems and

are shown to be highly efficient. Since some portion of problem-specific information is usually injected into a chromosome (which, in turn, leads to problem-specific genetic operators), such methods can not be used for different problems. However, the indirect GA approaches described above are generalised in some recent work.

In [20], Cowling et al. develop a GA-based hyperheuristic approach, called *hyper-GA*, and test it on a simple model for a real-world trainer scheduling problem. An indirect GA operates at a higher level and evolves a sequence of low-level heuristics from a given set. The low-level heuristics are then applied in the order they appear in the sequence to find a good solution of the problem instance. The set of low-level heuristics contains twelve problem-specific heuristics based on combinations of adding, swapping, and deleting events in the schedule. The chromosome for a hyper-GA represents a sequence of integers corresponding to low-level heuristics. The length of the chromosome is equal to the number of low-level heuristics in the set so that each heuristic could be possibly present exactly once in the chromosome. Thus, each individual in a hyper-GA population encodes a sequence of low-level heuristics and indicates which heuristics to apply and in what order. A hyper-GA uses a one-point crossover operator and a mutation operator which replaces the low level heuristics in randomly selected positions of the chromosome by other low level heuristics from the set. The hyper-GA presented in [20] produces significantly better solutions than individual low-level heuristics and outperforms the direct GA and memetic algorithm for all 5 instances of the problem both in the quality of the solutions and CPU time used. The analysis of the behaviour of hyper-GA reveals that the hyper-heuristic tends to change the range of low-level heuristics in chromosomes as the search progresses selecting more often the heuristics which lead to improved solutions.

Han et al. further improve hyper-GA in [39]. Since the optimal length of the chromosome for hyper-GA is unknown, they developed a mechanism that adaptively changes the chromosome length during the search. This mechanism allows hyper-GA to evolve the best combinations of low-level heuristics which may contain different number of heuristics. Indeed, in some cases it is reasonable to remove from the sequence the heuristic (or heuristics) which worsen the current solution hence making the chromosome shorter. In other situations, insertion of "good" heuristics into the chromosome may be necessary so that the chromosome becomes longer. The idea of the adaptive length chromosome in hyper-GA is embodied in [39] by introducing specific crossover and mutation operators which operate with groups of genes. A similar approach with variable length of the chromosomes is implemented by Han and Kendall in [37], where they develop a strategy which decides whether to make the chromosome longer or shorter (by means of applying different mutation operators) in order to maintain its length consistent with the average length of the chromosomes over previous generations. Another version of hyper-GA is considered in [38], where the length of the chromosome is regulated by making poorly performing genes tabu for a number of generations. Note that all versions of hyper-GA maintain

a high level of generality and have the potential to be applicable to a range of problems with only minor modifications. However, hyper-GA has been applied only to a simplified version of the trainer scheduling problem and no results have been reported for other problems.

Ross et al. [59] propose an interesting GA-based hyperheuristic approach where the fitness of a chromosome is determined by its ability to successfully solve different instances of the same problem. The approach is implemented for a one-dimensional bin-packing problem where many benchmark instances are available from the literature. A chromosome consists of a number of blocks (genes). Each block contains information about the instance of the problem state and low level heuristic associated with this instance. For a bin-packing problem, the information related to the problem state represents the proportions of the items of different sizes remaining to be packed. The genetic operators perform crossovers and mutations either at block level or inside blocks. Each chromosome from the population is tested on different problems from a *training* set in order to calculate its fitness. At every stage of a bin-packing process, the current state of the problem is compared against the instances encoded in the blocks of the chromosome, thus determining the block representing the closest instance. The low level heuristic associated with the latter instance is then applied to the actual problem state. The fittest chromosome after a specified number of generations is used to solve problems from a *test* set. Ross et al. [59] report that their GA-based approach achieves optimal solutions for most of the problem instances considered and outperforms each low level heuristic applied separately. Although the idea used in [59] can be applied when considering other problems, the approach has some significant limitations. First, it requires many problem instances to be included into training and test sets which are often unavailable for real-world problems. Second, it can be much more difficult to encode the problem state instance for problems with complex structures than for a relatively simple bin-packing problem, as well as to define a measure of distance between different problem states. Finally, the approach might be expected to be very slow for a range of complex real-world optimisation problems.

4.2 Other Metaheuristic-Based Hyperheuristics

A proven record of successful applications of GA-based hyperheuristics to various problems has founded an interest in using other metaheuristics as higher level heuristic selectors. Most of the relevant approaches have been developed over very recent years. The exception is an early publication of Storer et al. [62] where the authors study the effects of performing the search in two different search spaces which are alternatives to the commonly used solution space. The list of papers discussing hyperheuristics based on popular metaheuristic methods is given in Table 3.

In [62], Storer et al. consider minimising makespan in a job shop scheduling environment. They define two search spaces namely problem space and heuristic space as a basis for local search algorithms. The idea of the search in a problem space is to apply a base heuristic (for example, simple SPT dispatching rule

Table 3. Hyperheuristic based on non-GA metaheuristics

Approach	Papers	Details
Simulated annealing	Bai and Kendall in [4] Storer et al. [62] Soubeiga [61] Chakhlevitch [16]	Random selection of LLH; probabilistic acceptance criteria
Tabu search	Storer et al. [62]	Basic version
	Kendall and Mohd Hussin [47]	Basic version
	Kendall and Mohd Hussin [48]	Hybrids with hill-climbing and great deluge methods; random tabu durations
	Burke et al. [8], [10]	Constructive version
	Burke and Soubeiga [15]	Combines tabu search and reinforcement
	Burke et al. [13], [9]	learning; variable tabu list size
	Dowsland et al. [26]	
	Cowling and Chakhlevitch [18], [19]	Methods with different tabu list contents; tabu list size is either fixed or automatically adjusted
VNS	Qu and Burke [57]	Neighbourhoods of LLH sequences of different lengths are explored

for job shop problem) to a perturbed versions of the original problem (where processing times for operations are slightly modified) in order to generate alternative sequences of scheduled jobs. These solution sequences are evaluated using original data and the best solution is recorded. The main point of our interest in this work, however, is the concept of *heuristic space*, which is the basis for any hyperheuristic. In [62], the heuristic space contains strings of dispatching rules of a specified length, selected from the set of 6 rules commonly used in machine scheduling. The string of rules defines which rules and in which order should be called by a base heuristic (schedule generator) in the process of schedule construction when a decision about the operation to be scheduled next is required. Apart from random, hill-climbing and steepest descent methods, Storer et al. [62] study the performance of basic versions of popular metaheuristics in heuristic space. Simulated annealing, tabu search and genetic algorithm are applied for searching heuristic space and tested on a range of hard job shop scheduling problems. The authors report the consistency and high quality of results produced by these metaheuristic-based hyperheuristics and conclude that heuristic space search can be very "useful in providing fast solutions to very large problems".

Simulated annealing [1] and tabu search [32] approaches can be used to control the search in heuristic space in a similar manner as they manage the neighbourhoods of problem solutions in conventional Metaheuristics. In the context of a hyperheuristic, both algorithms decide at each iteration whether to accept or to reject the application of a particular low level heuristic to the current solution of

the problem, depending on the objective value which would result after applying the low level heuristic.

Chakhlevitch in [16] demonstrates that a simulated annealing hyperheuristic produces more consistent results across a range of instances of a relatively detailed model of a real-world trainer scheduling problem than a problem-specific version of a simulated annealing metaheuristic. Additionally, the hyperheuristic approach is less sensitive to the choice of initial solution for the problem than its metaheuristic counterpart. Other examples of simulated annealing based hyperheuristics can be found in [4] and [61]. We also refer to discussion of these methods in Section 2.

Hyperheuristics based on the tabu search metaheuristic have received increasing attention in recent publications. Kendall and Mohd Hussin [48] consider a simple tabu search based hyperheuristic for solving examination timetabling problem. Their hyperheuristic manages a set of 13 low level heuristics based on adding, moving, swapping and removing exams in the timetable. A low level heuristic becomes tabu as soon as it is applied to the current solution irrespective of whether it improves the solution or not. The tabu duration for each low level heuristic is short and fixed and there is no aspiration criterion. This means that a low level heuristic can not be applied while it remains tabu, even if it leads to the largest improvement among all low level heuristics. The best non-tabu low level heuristic is applied instead. Although such a simplified approach is never able to beat the best known results for a range of benchmark timetabling problems, it consistently produces good quality outcomes provided a sufficient amount of CPU time is available. In [47], Kendall and Mohd Hussin consider two more advanced versions of tabu search based hyperheuristic developed in [48]. In the first version, a low level heuristic which improves the previous best solution is applied repeatedly and becomes tabu only when it does not produce further improvements (tabu search hyperheuristic with hill climbing). The second version accepts the best non-improving and non-tabu low level heuristic only if it updates the solution within a certain boundary (the idea used in the great deluge algorithm, see [27]). In addition, random tabu durations from a given range are considered for both versions. The authors report further improvements to results obtained in [48]. Another tabu search based hyperheuristic approach for tackling examination timetabling problems is proposed by Burke et al. [8]. Instead of starting the search from the previously constructed initial solution, their method starts from a blank timetable and generates the sequences of low level heuristics which are used for step-by-step timetable construction. They use only two low level heuristics which represent two different ordering strategies widely used in examination timetabling. Tabu search is performed over a space of permutations of these two heuristics. The hyperheuristic outperforms both low level heuristics applied separately, losing, however, to problem-specific approaches on benchmark timetabling problems. A similar approach with six low level heuristics is considered in [10].

Burke and Soubeiga [15] employ tabu lists of poorly performing low level heuristics in a hyperheuristic approach to solving the nurse rostering problem.

In their method, low level heuristics compete against each other using rules based on the principles of reinforcement learning. There are 9 low level heuristics in the set which are ranked according to their performance during the hyperheuristic run. At the beginning of the search each low level heuristic receives zero score and the scores are dynamically changed as search progresses. Note that similar idea is used in the hyperheuristic developed by Nareyek [53] and in choice function hyperheuristics ([21] – [24]) which will be discussed in the next section. If the applied low level heuristic yields improvement to the current solution, its score is incremented (positive reinforcement), otherwise it is decreased on a specified number of points (negative reinforcement). However, such a scheme has a disadvantage of repetitive calls of the poorly performing low level heuristics with the highest scores (until the scores become low enough). The highest scores for such low level heuristics have been achieved due to improvements produced in the earlier stages of the search. In order to overcome this problem, each applied non-improving low level heuristic immediately becomes tabu and is released from the tabu list as soon as the current solution is changed by some other low level heuristic. Therefore, the size of the tabu list is variable and depends on the number of low level heuristics applied before the current solution is changed. The authors present results of a high quality for different instances of the nurse scheduling problem. They also claim the robustness of their approach across a range of instances of different problems. This claim is supported in [13] where the hyperheuristic is applied to the university course timetabling problem, outperforming two problem-tailored metaheuristics in terms of feasibility of solutions and showing competitiveness in terms of solution quality. In [9], tabu search based hyperheuristic approaches developed in [15] are adapted to solving multiobjective optimisation problems of space allocation and course timetabling. Finally, similar approach is used within simulated annealing framework to solve a complex shipper rationalisation problem (see [26]).

Cowling and Chakhlevitch [18], [19] present different versions of tabu search based hyperheuristics for the trainer scheduling problem. These hyperheuristics are designed to manage a large collection of (nearly 100) low level heuristics. The basic hyperheuristic employs a tabu list of recently called low level heuristics which have not improved the objective function. The algorithm greedily selects the best low level heuristic at each iteration of the search. If such a heuristic leads to an improved objective function value, it is always accepted and released from the tabu list if present; a non-improving heuristic is chosen only if it is not in the tabu list and immediately becomes tabu after its application. The authors test several versions of hyperheuristics with fixed and dynamically changed tabu list sizes as well as with different contents of tabu list such as recently applied non-improving low level heuristics and recently modified events. The results reported for tabu search hyperheuristics are advantageous to those obtained for other hyperheuristic methods considered in [18] and [19].

Qu and Burke in [57] develop a variable neighbourhood search (VNS) [40] hyperheuristic for the examination timetabling problem. As in [10], a timetable is generated by consecutively applying constructive low level heuristics in an

order specified by their sequence to schedule exams. The search is performed over a space of all possible sequences of low level heuristics of a given length. In [57], the neighbourhoods are defined by random replacement of two, three, four and five low level heuristics in a sequence. The hyperheuristic explores each neighbourhood for a specified number of iterations before switching to another neighbourhood. However, the results of [57] do not demonstrate this approach to be advantageous when compared to other hyperheuristics which use a single neighbourhood.

To conclude this section, we note that metaheuristic-based hyperheuristics have been tested on different real-world problems and shown to be very effective, even beating state-of-the-art problem-tailored methods on occasion. However, like traditional metaheuristic approaches, such hyperheuristics require fine tuning of parameters (temperature for simulated annealing, tabu list length or tabu tenure for tabu search, crossover and mutation rates for GA, etc.). Although hyperheuristics are often less sensitive to changes of these parameters, there is no guarantee that a hyperheuristic will work equally well for different problems using the same parameter settings. Recall that one of the main goals of a hyperheuristic is to provide a general framework for quickly producing solutions of a good quality for problems from different domains. The ideal hyperheuristic should be parameter-free (or nearly parameter-free) and easily applicable to a new problem without significant modifications and tuning. A few efforts have been undertaken to develop such methods by means of embedding learning techniques into hyperheuristics. We review hyperheuristics with learning in the next section.

5 Hyperheuristics with Learning

Hyperheuristics from this group employ various techniques for learning the historical performance of low level heuristics. A hyperheuristic selects a promising low level heuristic at each decision point based on the information about the effectiveness of each low level heuristic accumulated in earlier stages of its run (or in previous runs). Table 4 provides a list of publications together with a brief details of of the techniques used in this area.

One popular learning mechanism which has been employed in a hyperheuristic framework is based on the principles of reinforcement learning [44]. The general idea of such a technique is to "reward" improving low level heuristics at each iteration of the search and "punish" poorly performing ones by means of respectively increasing and decreasing their weights (scores) or probabilities of being selected. The weights of low level heuristics are adaptively changed as the search progresses and reflect the effectiveness of low level heuristics at any stage of the search.

Nareyek in [53] presents a weight adaptation method based on reinforcement learning. He investigates different schemes of selecting the promising heuristics from the set of alternatives during the search. Each heuristic has a weight assigned to it. The weight of a heuristic is changed as soon as a heuristic has

Table 4. Hyperheuristics with learning

Approach	Papers	Details
Reinforcement learning	Nareyek [53]	Weight adaptation methods
	Burke and Soubeiga [15]	LLH score adjustment within a tabu search framework
	Fisher and Thompson [31]	Adjustment of LLH selection probabilities
Choice function	Cowling et al. [21]- [24]	Three-component choice functions are used to keep track of the historical performance of LLHs
	Kendall et al. [49]	
	Soubeiga [61]	
Learning subsets of LLH	Chakhlevitch and Cowling [17]	Different learning criteria and strategies are used to choose effective LLH from a large set
	Chakhlevitch [16]	
Learning classifier system	Ross et al. [60]	Learning effective combinations of problem states and LLHs for their solving
Case based reasoning	Burke et al. [11]	LLHs suitable for modifying partial solutions of the problem are retrieved from the case base

been called and its performance has been evaluated. If the choice of the particular heuristic leads to improvement of the objective function, the weight of this heuristic increases, otherwise the weight decreases. The weights are bounded from above and from below. Nareyek considers different schemes for weight adaptation during the search and separates these schemes for the cases of improvement and deterioration. The current values of the weights express the information about the past experience of using the corresponding heuristics and depend on the region of the search space under exploration. The author presents two methods of selection of the heuristics based on their weights. The first one is the roulette-wheel approach where the heuristic is randomly selected with the probability proportional to its weight. The second method simply selects the heuristic with the maximum weight. A learning strategy (i.e. a hyperheuristic) combines three components: the weight adaptation scheme for the case of improvement, the scheme for the case of non-improvement, and heuristic selection method. The results of applying different strategies to two real-world optimisation problems (*Orc Quest* problem and the *Logistics Domain*) are reported. The hyperheuristic with the weight adaptation mechanism outperforms the stationary expert strategy even when the latter has a carefully selected combination of weights.

Other examples of hyperheuristics using principles of reinforcement learning include methods of Fisher and Thompson [31] and Burke and Soubeiga [15] (see also [13] and [9]) discussed in previous sections. Note that the former approach employs learning to adjust probabilities of selecting low level heuristics, while the latter uses the learning schemes similar to those in [53].

Cowling et al. in [21] introduce a hyperheuristic approach based on statistical ranking of low level heuristics. In this method, historical information about the

recent performance of low level heuristics is accumulated in a *choice function*. The selection of low level heuristic at each decision point depends on the current value of the corresponding choice function. They define the choice function as a "key to capturing the nature of the region of the solution space currently under exploration and deciding which neighbourhood (low-level heuristic) to call next, based on the historical performance of each neighbourhood". In [21], the choice function represents the weighted sum of the three components which reflect recent performance of each low-level heuristic, recent effectiveness of consecutive pairs of low-level heuristics, and the amount of time since the heuristic was last called, respectively. The first two components provide the intensification of the search while the third one is included for diversification. A good balance between intensification and diversification factors allows the hyperheuristic to explore the search space effectively. The weights of the components (denoted by α, β, δ respectively) express their relative importance in the choice function. The choice functions for low-level heuristics are recalculated at each iteration of the hyperheuristic. The general idea of the choice function is that the choice of an effective low-level heuristic at any given time may be stipulated by the recent successful application of the heuristic or by the effectiveness of this heuristic in combination with another heuristic, or, if the local optimum is reached, by the opportunity to redirect the search to a new region of the solution space. Choice function based hyperheuristics produce significantly better results for a simplified model of a real-world sales summit scheduling problem than those provided by the currently used scheduling system.

The limitations of the approach mentioned above are that it requires a warm-up period during which the heuristics should be selected randomly in order to initialise the values of the choice functions and that the weights α, β,and δ of individual components in the choice function should be manually tuned to achieve the best results. To overcome these limitations, Cowling et al. have developed an adaptive procedure that automatically adjusts the choice function's parameters during the search [22]. The method of parameter adjustment is to "reward" the improving heuristics and to "penalise" the non-improving heuristics ensuring that the best heuristics will be selected frequently and the worst ones will not be chosen very often. Such an adaptive procedure of parameter adjustment makes the hyperheuristic essentially parameter-free. The parameter-free hyperheuristic approach provides further improvements in the quality of the solutions of the sales summit scheduling problem (see [22]). The effectiveness and robustness of the approach are further investigated in [23] and [49] for the project presentation problem and in [24] for the nurse rostering problem. A detailed discussion and analysis of the choice function based hyperheuristics can be also found in [61].

Chakhlevitch and Cowling in [17] and Chakhlevitch in [16] consider the trainer scheduling problem and employ learning strategies embedded into peckish and tabu search based hyperheuristics in order to identify the subsets of the most effective low level heuristics in a large set. One of the reasons for introducing learning techniques is that, given a particular instance of the problem and a large collection of low level heuristics, it is difficult to predict in advance the behaviour

of different heuristics. Some low level heuristics may be particularly useful while other ones may bring no or little contribution to the solution process. Moreover, reducing the number of low level heuristics in the set may significantly speed up the search for a better solution. The authors consider two learning strategies. According to the first strategy, a hyperheuristic removes a certain number of the weakest low level heuristics after a fixed number of iterations and then continues its run with a reduced set. In the second strategy, low level heuristics with a poor performance are eliminated continuously during the hyperheuristic run until a required number of the best ones remains in the set. In [17] and [16], several selection criteria for low level heuristic ranking based on their ability to make changes to the current solution, frequency of calls by a hyperheuristic, frequency and magnitude of improvements are tested. The results of the experiments suggest that hyperheuristics with embedded learning strategies outperform hyperheuristics without learning given similar CPU time.

In [60], Ross et al. use a learning classifier system [65] to learn a set of rules for solving one-dimensional bin-packing problems. As in [59], the rule is a combination of a problem state and an associated low level heuristic (see discussion in subsection 4.1 of the GA-based method used in [59]). The learning classifier system works on binary representation of rules. The set of benchmark bin-packing problems is divided on two subsets used for training the learning classifier system and for testing the learned rules respectively. The method achieves similar results and has similar disadvantages to the GA-based approach in [59].

Burke et al. in [11] develop a hyperheuristic approach employing case based reasoning for low level heuristic selection and apply it to the examination timetabling problem. The approach has similarities to that presented in [60] which uses the learning classifier system. The timetable is constructed step-by-step by applying a low level heuristic to the partial solution at each step. The appropriate low level heuristic is retrieved from a *case base*. The case base is a collection of cases where each case describes possible partial solution and suggests a low level heuristic which has been previously found to be effective in dealing with such a partial solution. The cases in a base are picked up in a process of solving the problems from a training set. The partial solution in each case is represented by a list of features (properties) which is determined earlier at a knowledge discovery stage by a specific tabu search procedure. After the case base has been formed, it is tested on another set of problems (test set). At each step of timetable construction, the hyperheuristic identifies the case which is the closest to the current partial solution and applies the low level heuristic recorded in this case. The authors demonstrate in [11] that a hyperheuristic with case based low level heuristic selection consistently outperforms individual low level heuristics for a range of timetabling problems.

6 Other Generic Problem Solving Techniques

In this section we consider a range of AI approaches which are closely related to hyperheuristics. These approaches are aimed to providing a general methodology

Table 5. Generic methods related to hyperheuristics

Papers	Details
Gratch and Chien [34] Gratch et al. [35]	A statistical approach used to identify the best strategy (combination of LLHs) for solving problems from a given distribution
Minton [51], [52]	Expert system which generates efficient computer programs to solve constraint satisfaction problems
Fink [29] Gupta et al. [36] Petrovic and Qu [56] Burke et al. [14] Lagoudakis and Littman [50]	Various techniques to select a single heuristic from a set of possible alternatives
Randall and Abramson [58]	A generic problem solver based on the linked-list formulation of the problems

for solving various instances from a selected problem domain. The main goal is usually either to automatically select a good (ideally, the best) problem solving method (heuristic) from a list of possible alternatives or to learn a good strategy (combination of heuristics) which performs well over a distribution of problem instances. The final choice of the solution method or strategy is based on the historical performance of the alternatives on a training set of problems. Table 5 summarises recent developments in this area.

Gratch et al. [35] and Gratch and Chien [34] consider a statistical approach to adaptively solve the real-world problem of scheduling satellite communications. They develop a machine learning system that performs a hill-climbing search over a space of possible combinations of heuristic methods (called control strategies) and returns the most effective combination for the given problem domain. The performance of each control strategy is evaluated by means of statistical techniques. The heuristic scheduling algorithm makes its control decisions applying the corresponding heuristic from the control strategy at each decision point. The authors specify 5 decision points during the process of solving each problem instance with a set of possible low level heuristics to try at each decision point. The sample of the problems for each run of the system is formed by random selection of the specified number of problems from the distribution. The best strategy is determined from several runs of the system since the random selection of training problems may result in different learned strategies on different runs. The selected strategy is then used to solve all the problems from a given distribution. Gratch and Chien [34] report a significant improvement in the performance of their adaptive learning approach in comparison to the system which employs a human expert strategy. The learned control strategies outperform the expert strategy both in terms of CPU time required to produce a feasible schedule and the number of problems from the problem distribution solved within a specified resource bound.

Note that although the idea of the adaptive problem solving used in [35] and [34] is applicable to other problem domains, the approach may require significant modifications. Indeed, the adaptive problem solver employs domain-specific knowledge which is expressed not only in the structure of the control strategy, but in the method of exploration of the control strategy space as well. Another limitation of the solver is the problem of the local maxima due to the nature of the hill-climbing search. Finally, the statistical approach to adaptive problem solving implemented in [34] is computationally expensive since it requires a large number of training examples in order to evaluate the performance of the strategy.

Minton in [52] and [51] describes the expert system, Multi-Tac, for solving constraint satisfaction problems [5]. The objective of the system is to produce an efficient Lisp program tailored to particular problem and instance distribution. Since many combinatorial optimisation problems can be formulated as constraint satisfaction problems, the system can be used for solving problems of various natures. Multi-Tac specialises a set of generic heuristics for constraint satisfaction problems for a particular application and performs the search for the best combination of the domain-specific versions of these heuristics. This combination is then used to generate the problem-specific program. As for [34], Minton uses hill-climbing search over the space of heuristic combinations is performed and each combination is evaluated on a set of training instances. The system is tested on two well-known NP-hard problems and the resulting synthesised programs are shown to be competitive with the programs developed by human experts.

Fink [29] develops a statistical technique for automatic selection among available problem solving methods (heuristics) for a given problem instance. This technique is implemented in the framework of a sophisticated AI planning system and combines knowledge acquired from the past performance of the methods, for solving other problem instances from the same distribution, with exploration of new alternatives. Incremental learning is used for selection of the solving method. If the past performance data for all methods are available, a weighted random selection among the methods is performed, where a weight for each method represents the probability that the method is the best for a given problem instance (exploitation). If there is no previous data for some method, it is immediately selected and applied (exploration). The technique is tested on a large set of transportation problems and proved to be effective.

Several other approaches have been developed to learn a single best heuristic (strategy) from the set of possible alternatives for solving a range of problems. For example, Gupta et al. in [36] study the application of neural networks to selecting the best heuristic algorithm for a flowshop scheduling problem. Petrovic and Qu [56] and Burke et al. [14] employ case based reasoning to select a heuristic solving method for course timetabling problems. An approach based on reinforcement learning is proposed by Lagoudakis and Littman [50] to select the most efficient algorithm for solving large instances of simple problems of order statistic selection and sorting.

Randall and Abramson in [58] develop a general problem solver for combinatorial optimisation problems. Their solver is based on simulated annealing and tabu search metaheuristics. The crucial point of their system is a modelling representation for combinatorial optimisation problems. The authors introduce an alternative representation based on dynamic data structures, specifically multi-level linked lists. The linked list representation is well suited for many combinatorial optimisation problems. List modelling also allows for the elimination of many constraints which would typically appear in a traditional integer linear programming formulation of the problem since the range of possible values can be defined for the elements of the list. Finally, traditional local search moves like swapping, adding and repositioning components of the solution can be easily implemented within the list structure in terms of inserting and deleting elements in the list.

The problem solver has been tested by the authors on many instances of the benchmark combinatorial optimisation problems. The system has been able to produce optimal or close to optimal solutions in most occasions in a reasonable time. However, the approach has been applied only to relatively easy problems, which require only one level of sublists for their list-based formulation. Note that the majority of real-world problems are much more complicated and multi-level list structures may be needed for their representation. There is no evidence in [58] that the solver would perform well on such problems. Another significant drawback of the solver is a substantial amount of computer time required for parameter tuning.

7 Conclusions

Hyperheuristics represent an interesting direction in the development of generic solving techniques for combinatorial optimisation problems. Although general problem solving methods have received increased attention among researchers over the last two decades, most of the approaches presented in the literature have been designed to solve various instances of a particular problem rather than to be applied to a range of different problems. One of the main advantages of hyperheuristics over traditional problem-tailored approaches is their reapplicability and robustness across different problem domains. Development of problem-independent hyperheuristic approaches is an important and challenging task and research to date provides only a few promising initial steps in this direction.

In this chapter we have reviewed a wide spectrum of techniques which can be classified as hyperheuristics as well as a range of approaches closely related to them. In our review we have opted for a detailed analysis of the ideas lying behind different hyperheuristic approaches in order to identify their advantages and drawbacks. We can mention the following common limitations.

- Some hyperheuristic techniques make use of additional problem specific knowledge. For example, such knowledge can be used to describe the current state of the problem in order to select a suitable low level heuristic in

hyperheuristics employing learning classifier systems. In indirect GAs, a portion of problem-specific information is often injected into the chromosome.
- For many hyperheuristics, a significant amount of parameter tuning is required in order to find good parameter settings for a given problem.
- A large number of problem instances may be required for training and testing of the method in order to accumulate enough knowledge to make the right choice of low level heuristics. However, for many real-world problems the problem data are not easily available and randomly generated instances may not adequately represent the real distribution.
- Many hyperheuristic methods are only tested on a relatively simple benchmark problems for which the best solutions (often optimal) as well as effective low level heuristics are known in advance. There is no evidence that such hyperheuristics would be effective in more complex real-world situations.

Future research efforts in the area of hyperheuristics should be undertaken in order to overcome these limitations. Development of effective parameter-free hyperheuristics, methods for automatic parameter tuning in hyperheuristics, and creating new techniques with a clear boundaries between the hyperheuristic (higher level) and problem-specific (lower level) components are important research directions. Hyperheuristics should be tested on a wider range of real-world optimisation problems of different nature which would be the key to proving their suitability as a fundamental component of future generic optimisation software.

References

1. Aarts, E.H.L., Korst, J.H.M., van Laarhoven, P.J.M.: Simulated annealing. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimisation, pp. 91–120. John Wiley & Sons, Chichester (1997)
2. Adams, J., Balas, E., Zawack, D.: The shifting bottleneck procedure for job shop scheduling. Management Science 34, 391–401 (1988)
3. Ayob, M., Kendall, G.: A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In: Proceedings of the 2003 International Conference on Intelligent Technologies (InTech2003), Thailand, pp. 132–141 (2003)
4. Bai, R., Kendall, G.: An investigation of automated planograms using a simulated annealing based hyper-heuristic. In: Proceedings of the 5th Metaheuristics International Conference (MIC2003), Kyoto, Japan, August 23-25 (2003)
5. Brailsford, S., Potts, C., Smith, B.: Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research 119, 557–581 (1999)
6. Brelaz, D.: New methods to colour the vertices of the graph. Communications of the ACM 22, 251–256 (1979)
7. Brucker, P.: Scheduling Algorithms. Springer, Heidelberg (1995)
8. Burke, E., Dror, M., Petrovic, S., Qu, R.: Hybrid graph heuristics within a hyperheuristic approach to exam timetabling problems. In: Golden, B.L., Raghavan, S., Wasil, E.A. (eds.) The Next Wave in Computing, Optimisation and Decision Technologies. Conference 9th INFORMS Computing Society Conference, vol. 9, pp. 79–91. Springer, Heidelberg (2005)

9. Burke, E.K., Landa Silva, J.D., Soubeiga, E.: Multi-objective hyper-heuristic approaches for space allocation and timetabling. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solvers. Selected Papers from the 5th Metaheuristics International Conference (MIC 2003). Operations Research/Computer Science Interfaces Series, vol. 32, pp. 129–158. Springer, Heidelberg (2005)
10. Burke, E., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyper heuristic for timetabling problems. Technical Report NOTTCS-TR-2004-9, School of Computer Science and Information Technology, University of Nottingham (2004)
11. Burke, E., Petrovic, S., Qu, R.: Case based heuristic selection for examination timetabling. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), pp. 277–281. Orchid Country Club, Singapore (2002)
12. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyperheuristics: an emerging direction in modern search technology. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer Academic Publishers, Dordrecht (2003)
13. Burke, E., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9, 451–470 (2003)
14. Burke, E.K., MacCarthy, B.L., Petrovic, S., Qu, R.: Knowledge discovery in a hyper-heuristic for course timetabling using case-based reasoning. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 90–103. Springer, Heidelberg (2003)
15. Burke, E., Soubeiga, E.: Scheduling nurses using a tabu-search hyperheuristic. In: Kendall, G., Burke, E., Petrovic, S. (eds.) Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003), Nottingham, UK, pp. 197–218 (2003)
16. Chakhlevitch, K.: A hyperheuristic methodology for real-world scheduling. PhD Thesis, Department of Computing, University of Bradford, UK (2006)
17. Cowling, P.I., Chakhlevitch, K.: Choosing the Fittest Subset of Low Level Heuristics in a Hyperheuristic Framework. In: Raidl, G.R., Gottlieb, J. (eds.) EvoCOP 2005. LNCS, vol. 3448, pp. 23–33. Springer, Heidelberg (2005)
18. Cowling, P., Chakhlevitch, K.: Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC 2003), pp. 1214–1221. IEEE Press, Los Alamitos (2003)
19. Cowling, P., Chakhlevitch, K.: Using a large set of low level heuristics in a hyperheuristic approach to personnel scheduling. In: Dahal, K., Tan, K.C., Cowling, P.I. (eds.) Evolutionary Scheduling. Springer, Heidelberg (to appear, 2007)
20. Cowling, P., Kendall, G., Han, L.: An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: Proceedings of 2002 Congress on Evolutionary Computation (CEC 2002), pp. 1185–1190. IEEE Computer Society Press, Honolulu, USA (2002)
21. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
22. Cowling, P., Kendall, G., Soubeiga, E.: A parameter-free hyperheuristic for scheduling a sales summit. In: Proceedings of the Third Metaheuristic International Conference (MIC 2001), Porto, Portugal, pp. 127–131 (2001)

23. Cowling, P., Kendall, G., Soubeiga, E.: Hyperheuristics: a tool for rapid prototyping in scheduling and optimisation. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 1–10. Springer, Berlin (2002)
24. Cowling, P., Kendall, G., Soubeiga, E.: Hyperheuristics: a robust optimisation method applied to nurse scheduling. 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. 851–860. Springer, Heidelberg (2002)
25. Dorndorf, U., Pesch, E.: Evolution based learning in a job shop scheduling environment. Computers and Operations Research 22, 25–40 (1995)
26. Dowsland, K., Soubeiga, E., Burke, E.: Solving a shipper rationalisation problem with a simulated annealing based hyperheuristic. Technical Report NOTTCSTR-2004-1, School of Computer Science and Information Technology, University of Nottingham (2004)
27. Dueck, G.: New optimisation heuristics: the great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104, 86–92 (1993)
28. Fang, H.-L., Ross, P., Corne, D.: A promising hybrid GA/heuristic approach for open-shop scheduling problems. In: Cohn, A. (ed.) Proceedings of ECAI 1994: 11th European Conference on Artificial Intelligence, pp. 590–594. John Wiley, Chichester (1994)
29. Fink, E.: How to solve it automatically: selection among problem-solving methods. In: Proceedings of the 4th International Conference of AI Planning Systems, pp. 128–136. AAAI Press, Menlo Park (1998)
30. Fisher, H., Thompson, G.L.: Probabilistic learning combinations of local jobshop scheduling rules. In: Factory Scheduling Conference, May 10-12, 1961, Carnegie Institute of Technology (1961)
31. Fisher, H., Thompson, G.L.: Probabilistic learning combinations of local jobshop scheduling rules. In: Muth, J.F., Thompson, G.L. (eds.) Industrial Scheduling, pp. 225–251. Prentice Hall, Englewood Cliffs (1963)
32. Glover, F., Laguna, M.: Tabu search. Kluwer Academic Publishers, Norwell (1997)
33. Glover, F., Laguna, M.: Tabu search. In: Reeves, C.R. (ed.) Modern Heuristic Techniques for Combinatorial Problems, pp. 70–150. Blackwell Scientific Publications, Malden (1993)
34. Gratch, J., Chien, S.: Adaptive problem-solving for large-scale scheduling problems: a case study. Journal of Artificial Intelligence Research 4, 365–396 (1996)
35. Gratch, J., Chien, S., DeJong, G.: Learning search control knowledge for deep space network scheduling. In: Proceedings of the 10th International Conference on Machine Learning, Amherst, USA, pp. 135–142 (1993)
36. Gupta, J.N.D., Sexton, R.S., Tunc, E.A.: Selecting scheduling heuristics using neural networks. INFORMS Journal on Computing 12, 150–162 (2000)
37. Han, L., Kendall, G.: Guided operators for a hyper-heuristic genetic algorithm. In: Gedeon, T.D., Fung, L.C.C. (eds.) AI 2003. LNCS (LNAI), vol. 2903, pp. 807–820. Springer, Heidelberg (2003)
38. Han, L., Kendall, G.: An investigation of a tabu assisted hyper-heuristic genetic algorithm. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC 2003), pp. 2230–2237. IEEE Computer Society Press, Canberra, Australia (2003)

39. Han, L., Kendall, G., Cowling, P.: An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), pp. 267–271. Orchid Country Club, Singapore (2002)
40. Hansen, P., Mladenović, N.: Variable neighbourhood search: Principles and applications. European Journal of Operational Research 130, 449–467 (2001)
41. Hart, E., Ross, P.: A heuristic combination method for solving job-shop scheduling problems. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 845–854. Springer, Heidelberg (1998)
42. Hart, E., Ross, P., Nelson, J.: Solving a real-world problem using an evolving heuristically driven schedule builder. Evolutionary Computation 6, 61–80 (1998)
43. Hart, E., Ross, P., Nelson, J.: Scheduling chicken catching – An investigation into the success of a genetic algorithm on a real-world scheduling problem. Annals of Operations Research 92, 363–380 (1999)
44. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
45. Kendall, G., Mohamad, M.: Channel assignment in cellular communication using a Great Deluge hyper-heuristic. In: Proceedings of the 2004 IEEE International Conference on Networks (ICON 2004), Singapore, November 16-19 (2004)
46. Kendall, G., Mohamad, M.: Channel assignment optimisation using a hyperheuristic. In: Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS 2004), Singapore, December 1-3 (2004)
47. Kendall, G., Mohd Hussin, N.: Tabu search hyper-heuristic approach to the examination timetabling problem at University of Technology MARA. In: Burke, E., Trick, M. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 199–217. Springer, Heidelberg (2005)
48. Kendall, G., Mohd Hussin, N.: An investigation of a tabu search based hyperheuristic for examination timetabling. In: Kendall, G., Burke, E., Petrovic, S., Gendreau, M. (eds.) Multidisciplinary Scheduling: Theory and Applications, Selected papers from the 1^{st} Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003), pp. 309–328. Springer, Heidelberg (2005)
49. Kendall, G., Soubeiga, E., Cowling, P.: Choice function and random hyperheuristics. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), pp. 667–671. Orchid Country Club, Singapore (2002)
50. Lagoudakis, M.G., Littman, M.L.: Algorithm selection using reinforcement learning. In: Proceedings of the 17th International Conference on Machine Learning, pp. 511–518 (2000)
51. Minton, S.: Integrating heuristics for constraint satisfaction problems: a case study. In: AAAI Proceedings (1993)
52. Minton, S.: An analytic learning system for specializing heuristics. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence (1993)
53. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In: Resende, M., de Sousa, J. (eds.) Metaheuristics: Computer decision-making, pp. 523–544. Kluwer Academic Publishers, Dordrecht (2003)
54. Norenkov, I.: Scheduling and allocation for simulation and synthesis of CAD system hardware. In: Proceedings of EWITD 1994, East-West International Conference, Moscow, ICSTI, pp. 20–24 (1994)
55. Norenkov, I., Goodman, E.: Solving scheduling problems via evolutionary methods for rule sequence optimisation. In: Second World Conference on Soft Computing (WSC2) (June 1997)

56. Petrovic, S., Qu, R.: Case-based reasoning as a heuristic selector in a hyperheuristic for course timetabling problems. In: Proceedings of the 6th International Conference on Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies (KES 2002), Crema, Italy, pp. 336–340 (2002)
57. Qu, R., Burke, E.: Hybrid variable neighbourhood hyperheuristics for exam timetabling problems. In: Proceedings of the 6th Metaheuristics International Conference (MIC 2005), Vienna, Austria (2005)
58. Randall, M., Abramson, D.: A general meta-heuristic based solver for combinatorial optimisation problems. Computational Optimisation and Applications 20, 185–210 (2001)
59. Ross, P., Marín-Blázquez, J.G., Schulenburg, S., Hart, E.: Learning a procedure that can solve hard bin-packing problems: a new GA-based approach to hyperheuristics. 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. 2723, pp. 1295–1306. Springer, Heidelberg (2003)
60. Ross, P., Schulenburg, S., Marín-Bl ázquez, J.G., Hart, E.: Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 942–948. Morgan Kaufmann, San Francisco (2002)
61. Soubeiga, E.: Development and application of hyperheuristics to personnel scheduling. PhD Thesis, Department of Computer Science, University of Nottingham, UK (2003)
62. Storer, R.H., Wu, S.D., Vaccari, R.: Problem and heuristic search space strategies for job shop scheduling. ORSA Journal on Computing 7, 453–467 (1995)
63. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
64. Terashima-Marín, H., Ross, P., Valenzuela-Rendón, M.: Evolution of constraint satisfaction strategies in examination timetabling. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), pp. 635–642. Morgan Kaufmann, San Francisco (1999)
65. Wilson, S.W.: Classifier systems based on accuracy. Evolutionary Computation 3, 149–175 (1995)

Self-Adaptation in Evolutionary Algorithms for Combinatorial Optimisation

James E. Smith

School of Computer Science, University of the West of England at Bristol, UK
james.smith@uwe.ac.uk

Summary. It is well known that the choice of parameter settings for meta-heuristic algorithms has a dramatic impact on their search performance and this has lead to considerable interest in various mechanisms that in some way attempt to automatically adjust the algorithm's parameters for a given problem. Of course this raises the spectre of unsuitable parameters arising from a poor choice of learning/adaptation technique. Within the field of Evolutionary Algorithms, many approaches have been tried, most notably that of "Self-Adaptation", whereby the heuristic's parameters are encoded alongside the candidate solution, and acted on by the same forces of evolution. Many successful applications have been reported, particularly in the sub-field of Evolution Strategies for problems in the continuous domain. In this chapter we examine the motivation and features necessary for successful self-adaptive learning to occur. Since a number of works have dealt with the continuous domain, this chapter focusses particularly on its aspects that arise when it is applied to combinatorial problems. We describe how self-adaptation may be use to control not only the parameters defining crossover and mutation, but also how it may be used to control the very definition of local search operators used within hybrid evolutionary algorithms (so-called memetic algorithms). On this basis we end by drawing some conclusions and suggestions about how this phenomenon might be translated to work within other search metaheuristics.

Keywords: Self-adaptation, evolutionary algorithms, memetic algorithms, self-adapted parameters, self-adapted operators.

1 Introduction: What Is Self-Adaptation

It is well known that the performance of most search heuristics is dependent on an appropriate choice of the parameters governing how new candidate solutions are generated and used by the algorithm. Details such as the cooling schedule in Simulate Annealing, or the length of the Tabu List are obvious examples of this. The idea of a search heuristic that could automatically adjust its parameters has generated considerable interest amongst researchers. The general idea is that by reducing the number of predefined parameters, the likelihood of poor performance arising from an unsuitable choice is reduced, and so more robust algorithms may be developed. Of course this instead raises the spectre of unsuitable parameters arising from a poor choice of learning/adaptation technique instead. Within the field of Evolutionary Algorithms, many approaches

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 31–57, 2008.
springerlink.com

have been tried, most notably that of "Self-Adaptation", whereby the heuristic's parameters are encoded alongside the candidate solution, and acted on by the same forces of evolution. Many successful applications have been reported, particularly in the sub-field of Evolution Strategies for problems in the continuous domain. In this chapter we examine the motivation and features necessary for successful self-adaption to occur.

This then raises a first question: What is Self-Adaptation? Informally, and (deliberately) rather vaguely, it is a property of natural and artificial systems that allows them to control the way in which they adapt to changing environments. In order to answer this question more fully, we can pose a number of related questions such as:

- Why should we be interested?
- When does it happen, and why?
- What might it be able to do?
- Does it only apply to Evolutionary Algorithms?

The rest of this chapter attempts to answer these questions. Since concept of Self-Adaptation originated within the field of Evolutionary Computation (EC) the 1970s, and this is where most of the subsequent research has been performed, most of the discussion that follow will be couched in the terminology of that field. However as we shall see the working definition that we define for the Self-Adaptation could in principle be applied to many other types of optimisation meta-heuristic. Furthermore, since the subject of self-adaptation of the parameters controlling search in continuous spaces has been extensively tackled elsewhere (e.g. [15, 51, 60]) we shall concentrate on the self-adaptation of parameters for combinatorial search problems, since this raises some interesting additional issues.

The rest of this chapter proceeds as follows: Section 2 answers the first question, focussing on why adaptive schemes are needed, building from there to create a taxonomy of possible adaptation schemes, and finishing with a list of some features that we consider an algorithm should possess for it to be classed as self-adaptive. Section 3 begins by describing how self-adaptation was first introduced by Schwefel and colleagues for adapting the step-sizes controlling mutation in continuous spaces. It then moves on to describe how these ideas have further developed for EC algorithms using discrete encodings, and some of the extra insights that have been derived as a result of this. Section 4 describes how self-adaptation has been used to control recombination, the other principle variation operator in EC, and in Section 5 we draw together the conclusions from these two areas of experimental activity, and place them together with some theoretical ideas that have been developed. Having by this stage hopefully answered the first three of our original questions, we end by turning our attention to the fourth, and presenting results from a system in which Self-Adaptation is used to define and generate Local Search operators within the context of a Memetic Algorithm.

2 Finding Appropriate Parameter Settings in Evolutionary Algorithms

Typically the process of designing an Evolutionary algorithm (EA) for a specific problem begins by considering the space of possible solutions to that problem and deciding upon a representation for the solutions. This then leads naturally to the choice of variation operators (e.g. recombination and mutation) that will be used to generate new solutions from the current population. A choice must also be made of parent selection operator (to decide how likely members of the population are to be used as inputs to the variation operators), and survival scheme (to decide how the next generation is to be created from the current one and outputs of the variation operators). For instance, for a simple Genetic Algorithm (GA) it might be decided to use binary representation, uniform crossover, bit-flip mutation, tournament selection, and generational replacement. For a full specification, however, further details have to be given, for instance, the population size, the probability of mutation p_m and crossover p_c, and the tournament size. These data – called the algorithm parameters or strategy parameters – complete the definition of the EA and are necessary to produce an executable version. The values of these parameters greatly determine whether the algorithm will find an optimal or near-optimal solution, and whether it will find such a solution efficiently. Choosing the right parameter values is, however, a hard task.

Globally, we distinguish two major forms of setting parameter values: **parameter tuning** and **parameter control**. By parameter tuning we mean the commonly practised approach of finding good values for the parameters *before* the run of the algorithm and then running the algorithm using these values, which remain fixed during the run. Later on in this section we give arguments that any static set of parameters having the values fixed during an EA run seems to be inappropriate. Parameter control forms an alternative, as it amounts to starting a run with initial parameter values that are changed *during* the run.

Parameter tuning is a typical approach to algorithm design. Such tuning is done by experimenting with different values and selecting the ones that give the best results on the test problems at hand. However, the number of possible parameters and their different values means that this is a very time-consuming activity. The technical drawbacks to parameter tuning based on experimentation can be summarised as follows:

- Parameters are not independent, but trying all different combinations systematically is practically impossible.
- Since these are stochastic algorithms, it is necessary to perform multiple runs for each combination in order to properly establish statistically significant differences.
- Therefore the process of parameter tuning is time consuming, even if parameters are optimised one by one, regardless of their interactions.
- For a given problem the selected parameter values are not necessarily optimal, even if the effort made for setting them was significant.

This picture becomes even more discouraging if one is after a "generally good" setup that would perform well on a range of problems or problem instances. During the history of EAs considerable effort has been spent on finding parameter values (for a given type of EA, such as GAs), that were good for a number of test problems. A well-known early example is that of [18], determining recommended values for the probabilities of single-point crossover and bit mutation on what is now called the De Jong test suite of five functions. About this and similar attempts [28,55], it should be noted that genetic algorithms used to be seen as robust problem solvers that exhibit approximately the same performance over a wide range of problems [26, page 6]. The contemporary view on EAs, however, acknowledges that specific problems (problem types) require specific EA setups for satisfactory performance [10]. Thus, the scope of "optimal" parameter settings is necessarily narrow. There are also theoretical arguments that any quest for generally good EA, thus generally good parameter settings, is lost a priori, for example the No Free Lunch theorem [84].

As hinted above, there is a perhaps more fundamental drawback of the parameter tuning approach. Recall how we defined it: finding good values for the parameters before the run of the algorithm and then running the algorithm using these values, *which remain fixed during the run*. However, a run of an EA is an intrinsically dynamic, adaptive process. The use of rigid parameters that do not change their values is thus in contrast to this spirit. Additionally, it is intuitively obvious, and has been empirically and theoretically demonstrated, that different values of parameters might be optimal at different stages of the evolutionary process [4,5,6,17,33,36,56,60,62,73,74,79,81].

This line of reasoning leads us to the realisation that rather than seeking some (near)-optimal "sweet-spot" within the space of all possible parameter values, we are actually concerned with finding a trajectory through that space that our algorithm will follow. Moreover, when we consider that EAs are a population-based heuristic, and that not all members of the population need have the same parameter settings, it becomes clear that it is more accurate to describe the progress of heuristic search as a cloud of points following a trajectory through space. For many algorithms this cloud will condense to a single point, and for static algorithms the position will remain fixed,but in general the cloud will move over time as different combinations of settings are tried or discarded at various stages in the progress of search. This happens under the influence of what we will term an update function. Most commonly, the progress of the search is monitored, e.g., by looking at the performance of operators, the diversity of the population, and so on. The information gathered by such a monitoring process is used as feedback for adjusting the parameters.

Various authors have attempted to classify the mechanisms which govern the trajectory through parameter space e.g. [2,20,75,77]. Here we will use the definitions synthesized from these in [21] and identify the three following approaches.

- **Deterministic parameter control.** This takes place when the value of a strategy parameter is altered by some deterministic rule. This rule modifies

the strategy parameter in a fixed, predetermined (i.e., user-specified) way without using any feedback from the search. Usually, a time-varying schedule is used, i.e., the rule is used when a set number of generations have elapsed since the last time the rule was activated.

- **Adaptive parameter control.** This takes place when there is some form of feedback from the search that serves as inputs to a mechanism used to determine the direction or magnitude of the change to the strategy parameter. The assignment of the value of the strategy parameter may involve credit assignment, so that the updating mechanism can distinguish between the merits of competing strategies based on the quality of solutions they produce. Although the subsequent action of the EA may determine whether or not the new value persists or propagates throughout the population, the important point to note is that the updating mechanism used to control parameter values is externally supplied, rather than being part of the "standard" evolutionary cycle.
- **Self-adaptive parameter control.** The idea of the evolution of evolution can be used to implement the self-adaptation of parameters (see [8] for a good review). Here the parameters to be adapted are encoded into the chromosomes and undergo mutation and recombination. The better values of these encoded parameters lead to better individuals, which in turn are more likely to survive and produce offspring and hence propagate these better parameter values. This is an important distinction between adaptive and self-adaptive schemes: in the latter the mechanisms for the credit assignment and updating of different strategy parameters are entirely implicit, i.e., they are the selection and variation operators of the evolutionary cycle itself.

This terminology leads to the taxonomy illustrated in Fig. 1.

Building on this categorisation, we can distinguish between two different types of adaptive algorithm, and further define what we mean by self-adaptation when we consider the evidence used for determining the change of parameter value [61,75]. Most commonly, the progress of the search is monitored, e.g., by looking at the performance of operators, the diversity of the population, and so on.

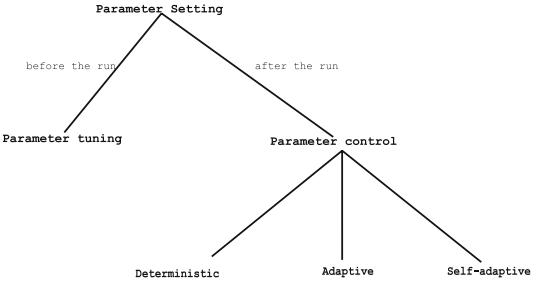


Fig. 1. Global taxonomy of parameter setting in EAs

From this perspective, we can make further distinction between the following two cases:

- We speak of **absolute evidence** when the value of a strategy parameter is altered by some rule that is applied when a predefined event occurs. The difference from deterministic parameter control lies in the fact that in deterministic parameter control a rule fires by a deterministic trigger (e.g., time elapsed), whereas here feedback from the search is used. For instance, the rule can be applied when the measure being monitored hits a previously set threshold – this is the event that forms the evidence. Examples of this type of parameter adjustment include increasing the mutation rate when the population diversity drops under a given value [50], changing the probability of applying mutation or crossover according to a fuzzy rule set using a variety of population statistics [48], and methods for resizing populations based on estimates of schemata fitness and variance [76]. Such mechanisms require that the user has a clear intuition about how to steer the given parameter into a certain direction in cases that can be specified in advance (e.g., they determine the threshold values for triggering rule activation). This intuition may be based on the encapsulation of practical experience, data-mining and empirical analysis of previous runs, or theoretical considerations (in the order of the three examples above), but all rely on the implicit assumption that changes that were appropriate to make on *another* search of *another* problem are applicable to *this* run of the EA on *this* problem. It is worth noting that many so-called self-adaptive algorithms in fact rely on this sort of evidence as inputs to some externally defined updating function, so do not fall into the definition of self adaptation offered by this article.
- In the case of using **relative evidence**, parameter values are compared according to the fitness of the offspring that they produce, and the better values get rewarded. The direction and/or magnitude of the change of the strategy parameter is not specified deterministically, but relative to the performance of other values, i.e., it is necessary to have more than one value present at any given time. Here, the assignment of the value of the strategy parameter involves credit assignment, and the action of the EA may determine whether or not the new value persists or propagates throughout the population. As an example, consider an EA using several different crossovers with crossover probabilities adding up to 1.0 and being reset based on the their performance measured by the quality of offspring they create. Such methods may be controlled adaptively, typically using "bookkeeping" to monitor performance and a user-supplied update procedure [17,37,58], or self-adaptively [5, 22, 47, 59, 72, 77] with the selection operator acting indirectly on operator or parameter frequencies via their association with "fit" solutions.

2.1 Features of Self-Adaptation

The rather broad discussion above permits identification of the following features that algorithm should possess in order to be classed as self-adaptive:

- The algorithm should exhibit automatic control of operators or their parameters, via the action of the evolutionary processes both in the space of possible solutions, and in the space of combinations of parameters. This means that each individual encodes for its own parameters.
- In order for the self-adaptation to be effective, there must be a range of different parameter sets present, so as to give evolution some diversity to work with. The range of values present will depend on the action of selection on the combined genotypes and the action of genetic operators (e.g. mutation) on those encoded values. This has some implications which will be explored later in this article.
- Finally, in order for successful evolution to occur there need to be links between the encoded parameter (which might be stochastic e.g. step sizes) and the subsequent change in problem encoding (hence some interest in derandomised mutations (e.g. [31]). There must also be links between the action of the operator and a subsequent change in fitness of the encoded solution, and between the individual fitness and the action of selection.

3 Self Adaptation of Mutation Operators

3.1 The Origins: Self Adaption in Evolution Strategies

Evolution Strategies (ES) are typically used in continuous search spaces where a solution with l components may be represented as a vector $\bar{x} \in \mathcal{R}^l$. The mutation operator is based on a normal (Gaussian) distribution requiring two parameters: the mean ξ and the standard deviation σ. In practice, the mean ξ is always set to zero, and so the vector $\bar{x}$ is mutated by setting each $x_i' = x_i + N(0, \sigma)$, where $N(0, \sigma)$ denotes a random number drawn from a Gaussian distribution with zero mean and standard deviation σ and this is done independently for each component $i : 1 \leq i \leq l$. By using a Gaussian distribution here, small mutations are more likely than large ones. The overall scale of the most probable mutations is governed by the value of σ, which is why it is commonly referred to as the "step size". It is intuitively obvious that for efficient progress the most beneficial step size will depend on how far the current solutions are from the global optimum.

The earliest ES used a single solution and offspring (the so-called (1+1) model) and Rechenberg invented his famous "1:5" heuristic rules for adapting step sizes according to the relative frequency with which improved solutions are found. As more sophisticated population models were developed by Schwefel and co-workers in the 1970's [59,60], other mechanisms became possible, and Self-Adaptation was identified as a successful method. Nowadays the very basis of self-adaptation in ES is that the step sizes are also included in the chromosomes and they themselves undergo variation and selection. In the simplest case we would have one step size that applied to all the components x_i and candidate solutions of the form $\langle x_1, \ldots, x_n, \sigma \rangle$. Mutations are then realised by replacing $\langle x_1, \ldots, x_n, \sigma \rangle$ by $\langle x_1', \ldots, x_n', \sigma' \rangle$. To obtain σ', the mutated value of the step-size σ, it is multiplied by a random variable drawn each time from a normal

distribution with mean 0 and standard deviation τ. Since $N(0,\tau) = \tau \cdot N(0,1)$, the full mutation mechanism is:

$$\sigma' = \sigma \cdot e^{\tau \cdot N(0,1)}, \tag{1}$$

$$x_i' = x_i + \sigma' \cdot N_i(0,1). \tag{2}$$

In these formulas $N(0,1)$ denotes a draw from the standard normal distribution, and since step-sizes very close to zero will have on average a negligible effect, a boundary rule is then applied to maintain σ at or above a small threshold value. The proportionality constant τ is an external parameter to be set by the user. It is usually inversely proportional to the square root of the problem size. It can be interpreted as a kind of learning rate, as in neural networks. Bäck [7] explains the rationale for this way of adapting σ as follows:

- Smaller modifications should occur more often than large ones.
- Standard deviations have to be greater than 0.
- The median (0.5-quantile) should be 1, since we want to multiply the σ.
- Mutation should be neutral on average. This requires equal likelihood of drawing a certain value and its reciprocal value, for all values.

The lognormal distribution satisfies all these requirements. What is important here is that the mutation step sizes are not set by the user; rather the σ is coevolving with the solutions (the $\bar{x}$ part). In order to achieve this behaviour it is essential to modify the value of σ first, and then mutate the x_i values with the new σ value. The rationale behind this is that a new individual $\langle \bar{x}', \sigma' \rangle$ is effectively evaluated twice. Primarily, it is evaluated directly for its viability during survivor selection based on $f(\bar{x}')$. Second, it is evaluated for its ability to create good offspring. This happens indirectly: a given step size evaluates favourably if the offspring generated by using it prove viable (in the first sense). Thus, an individual $\langle \bar{x}', \sigma' \rangle$ represents both a good $\bar{x}'$ that survived selection and a good σ' that proved successful in generating this good $\bar{x}'$ from $\bar{x}$.

The alert reader may have noticed that there is an important underlying assumption behind the idea of using varying mutation step sizes. Namely, we assume that under different circumstances different step sizes will behave differently: some will be better than others. These "circumstances" can be given various interpretations. For instance, we might consider "time" and distinguish different stages within the evolutionary search process and expect that different mutation strategies would be appropriate in different stages. Self-adaptation can then be a mechanism adjusting the mutation strategy as the search is proceeding. Alternatively, we can consider "space" and observe that the local vicinity of an individual, i.e., the shape of the fitness landscape in its neighbourhood, determines what good mutations are: those that jump into the direction of fitness increase. Assigning a separate mutation strategy to each individual, which coevolves with it, opens the possibility to learn and use a mutation operator suited for the local topology.As ever-more complicated problems have been considered it has become apparent that it is often not appropriate to be traversing each dimension of the search space at the same rate. This can be simply achieved

by encoding and adapting a separate step-size for each dimension. In fact for many functions it is useful to be able to identify and exploit search directions which do not lie parallel to any of the axes (dimensions), and this has lead to the development of models which include a covariance matrix describing how the likely moves in different dimensions are coupled to one another. However it must be borne in mind that although these arguments are plausible for continuous search space they do not transfer as easily to the combinatorial domain.

The central claim within ES is that self-adaptation works. Over the last decades much experience has been gained over self-adaptation in ES. The accumulated knowledge has identified necessary conditions for self-adaptation:

1. Population size $\mu > 1$ so that different strategies are present.
2. Generation of an offspring surplus: $\lambda > \mu$.
3. Not too strong selective pressure (as a heuristic: $\lambda/\mu = 7$, e.g., (15,100))
4. (μ, λ)-selection, where all parents are discarded are replaced by a subset of the best offspring, to guarantee extinction of poorly adapted individuals and strategies.
5. Recombination should also be used on the also on strategy parameters (especially intermediate recombination).

In addition to a wealth of experimental evidence, showing that an ES with self-adaptation outperforms the same ES without self-adaptation, there are also theoretical results backing up this claim. In fact, theoretical and experimental results can neatly complement each other in this area if for a (simple) objective function $f : \mathcal{R}^n \rightarrow \mathcal{R}$ theoretically optimal mutation step sizes can be calculated. Note that the problem and the algorithm must be simple to make the system tractable, since for a complex problem and/or algorithm a theoretical analysis is infeasible. Optimal mutation step sizes need to be defined in the light of some performance criteria, e.g., progress rate during a run. If experimentally obtained data show a good match with the theoretically derived values, then we can conclude that self-adaptation works in the sense that it is able to find the near-optimal step sizes. For reasons of tractability, the theoretical analysis of self-adaptation in Evolution Strategies has concentrated on looking at different forms of local topology such as hyperspheres and ridges, but this work is now well advanced and documented. Therefore we will not dwell on it in this chapter but refer the interested reader to works such as [11,15,31,30,51].

3.2 Self-Adaptation of Mutation for Discrete Encodings

A self-adaptive mechanism for controlling mutation in a bit-string GA is given by Bäck [4]. This technique works by extending the chromosomes by an additional 20 bits that together encode the individuals' own p_m. Mutation then works by:

1. Decoding these bits first to p_m
2. Mutating the bits that encode p_m with mutation probability p_m
3. Decoding these (changed) bits to p'_m
4. Mutating the bits that encode the solution with mutation probability p'_m

This approach is highly self-adaptive since even the rate of variation of the search parameters is given by the encoded value, as opposed to the use of an external parameter like τ in Eq. (1). Bäck's results showed that truncation selection was preferable (the equivalent of (μ, λ) selection with $\mu < \lambda$, but although he experimented with the use of multiple mutation rates for different encoded solution variables, the results were less conclusive.

Following Bäck's seminal work, several authors have experimented with self-adaptation of mutation rates in GAs (see for example, [12,24,34,49,74]. Design decisions that must be addressed with this approach are the choice of representation for the strategy parameter and, related to this, the means by which the strategy parameter is itself varied to allow adaptation to occur. Bäck's early work remained close to the traditional interpretation of a GA and used a binary encoding of the strategy parameters with corresponding bitwise mutation. Smith's work in the context of steady-state GAs examined a number of possibilities [74], and more current thinking is that a real-valued representation is preferable. This then allows the use of lognormal adaptation of strategy parameters as per Eq.(1).

However, subsequent work, both empirical [25,49] and theoretical [53], has shown that self-adaptation schemes which adapt too quickly can lead to premature convergence to low step sizes, with the search getting 'stuck' at local optimum. This has lead to an interest in alternative variation schemes. In order to investigate some of these from a theoretical perspective, Smith introduced a dynamical systems model of a GA with self-adaptation of mutation rates in [62]. The model is used to predict mean fitness of an evolving population over time. In order to reduce the algorithm to its bare essentials, with the added benefit of making the mathematics computationally tractable, there are two key differences between the model and the self-adaptive GAs just described. Firstly, rather than using a binary or real-valued representation, strategy parameters are represented by a single allele from a fixed, small alphabet of size q (a value of $q = 10$ was used in the original paper). A consequence of this is that the mutation rate attached to an individual can only take on one of q possible values, as opposed to the large or effectively infinite number available with binary or real-valued representations. Secondly, because of the discrete nature of the strategy parameter representation, the lognormal scheme cannot be used to vary the strategy parameters. The initial analysis used a scheme where a strategy parameter is modified with uniform probability z to one of the q possible alleles, which means that the behaviour of each of the strategy parameters can be modeled as a Markov chain, and the transition matrix associated with the mutation of the parameters has entries of the form:

$$P_{i,i'} = \begin{cases} z/q & i' \neq i \\ 1 - z/q & i' = i \end{cases} \tag{3}$$

The results of this analysis showed that even using a simple GA, with fitness-proportionate selection (i.e. without the offspring surplus previously thought to be necessary) it is possible to observe self-adaption even in dynamic environments. This can be evidenced by one experiment where after a fixed number of generations the fitness function was inverted from OneMax to ZeroMax. The

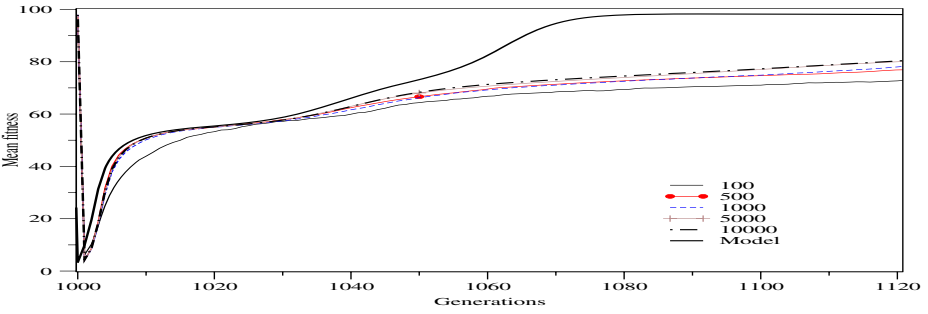


Fig. 2. Evolution of Mean Fitness After Environmental Change:Predicted vs. Empirical Results for different sized populations

mathematical model was used to predict both the mean fitness of the population as a function of time, and the proportions of the population using different mutation rates, and these were compared to experimental results from a GA with population size 1000.

Figure 2 illustrates the results of these experiments, concentrating on the period immediately after the transition. As can be seen there is a good match between the prediction and the observed behaviours, especially in the first thirty or so generations.

The patterns of (predicted and observed) evolved behaviour starting from a converged population are very different to those with the initial random population. This can be explained by examining the proportions of the population falling into the different mutation classes as shown in Figures 3 (predicted) and 4 (observed).

Initially after the transition, mutations are on average beneficial, and so once re-introduced by chance, individuals with the highest mutation rates attached start to take over the population. Note that under this simple model of self-adaptation this re-introduction by mutation occurs at a constant rate, which can happen faster than under a lognormal adaptation scheme (unless τ is set unusually high. However once the mean fitness has passed 50%, then on average mutations will be deleterious, and so there is a phase transition, and individuals with lower mutation rates attached have a selective advantage. Inspection of Figures 3 and 2 shows that this happens around 25 - 50 generations after the change. It is notable that the empirical and theoretical results are virtually identical up to this point for all population sizes over 100. From Figures 3 and 4 it can be seen that with a finite population the lowest mutation class does not takeover the population as much as is predicted after the phase transition. It has been suggested above that this is because the model predicts the early generation of individuals with high fitness, for which low mutation rates are selective advantageous, whereas the effects of a finite population mean that a more gradual evolution of fitness occurs, with correspondingly higher mutation rates.

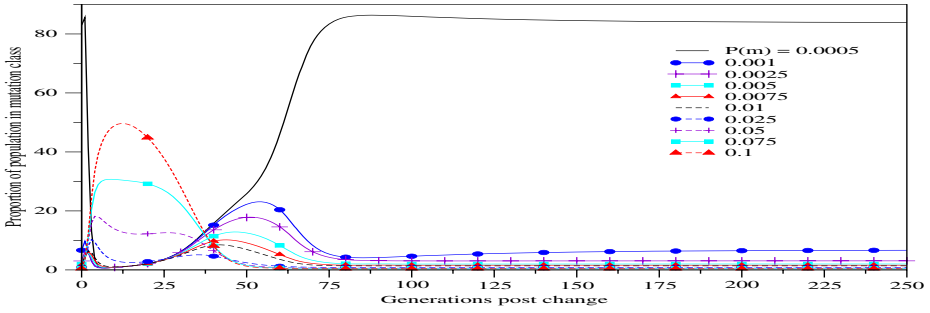


Fig. 3. Predicted Evolution of Proportions of Population in Different Mutation Classes After Environmental Change

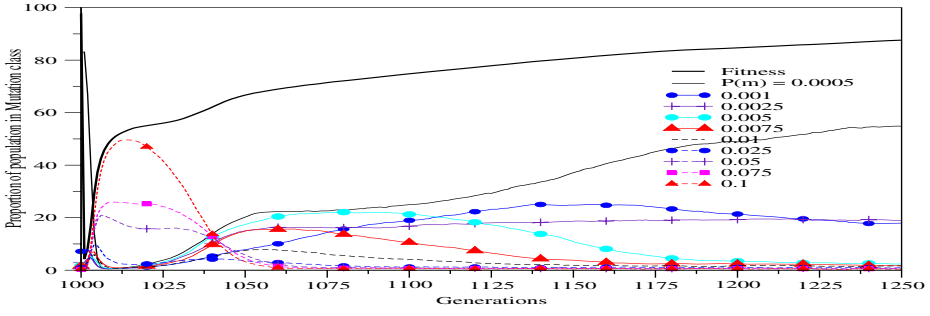


Fig. 4. Observed Evolution of Mean Fitness and Mutation Classes After Environmental Change, Population 1000

These results suggests that there is a limit to the rate of environmental change which the self adaptive algorithm can respond to, this limit being related to the time needed for this phase transition to occur. This time will depend on the selection pressure, but also on the Innovation Rate z, since this determines both the background proportions of (initially) sub-optimal high mutation rates in the population prior to the change, and also the rate at which lower mutation rates are re-introduced into the population.

Following the success of these models to correctly predict self-adaptive behaviour even in such a simple system, Stone and Smith compared the optimisation performance of the "discrete" scheme to algorithms using a continuous (i.e. real-valued) mutation parameter with log-normal schemes [80]. As noted above, the ES literature suggests that values of $\tau = c/\sqrt{n}$, for some constant c is suitable for a n-dimensional problem, but of course in the case of a GA a more appropriate measure is the length of the string in bits. It is not apparent that the rule can be directly mapped from ES to GA representations. Glickman and Sycara [24] use a value of $\tau = 0.1$ for a string of length 1000. This corresponds to a value of $c = 3.16$. In contrast, Hinterding, Michalewicz and Peachey [34] use

the lower fixed value of $\tau = 0.013$ in their self-adaptive GA. Stone and Smith tried values for c of 0.5, 1, 2, 3 and fixed rates for τ of 0.013 and 0.02. For the former values the problem length (in bits) is taken into account when arriving at the actual rates used, whereas the latter two are fixed rates across all functions. Table 1 shows the results obtained with $(100, 500)$ GA on a range of problem with very different characteristics in terms of modality, deception, plateaus, etc. As can be seen, the discrete scheme is consistently more effective at locating the global optimum across functions with a wide range of characteristics, although this comes at some expense of speed. The authors provide an explanation for this in terms of the diversity of mutation strategies that are present in the population at any time, and how this provides scope for evolution and for escaping local optima.

Table 1. Comparison of Optimisation Results for discrete self-adaptation of mutation rates vs. continuous scheme with log-normal adaptation. AES indicates mean time to locate the global optimum on successful runs, and SR indicates number of runs (out of 50) on which this was achieved. Full experimental details may be found in [80].

			Discrete Self-Adaptation			Continuous Self-Adaptation			
Function	length	Max. gens.	AES	SR	z	AES	SR	τ	c
OneMax	128	1000	68	50	0.05	66	50	0.063	2.00
Rastrigin's	64	5000	100	50	1	84	34	0.013	0.10
Deb's Deceptive	32	2500	1205	21	1	20	8	0.020	0.11
Matching Bits	128	10000	510	49	1	231	40	0.020	0.10
R1 Royal Road	64	10000	326	50	1	159	50	0.013	0.10

More recently Smith [66] extended his theoretical analysis to model different ways in which the attached mutation rates should be changed. This permitted him to study schemes where the value of z is not fixed as it is in the simple "discrete" scheme, but rather depends on the value of the encoded mutation rate - as it does for Bäck's approach and also log-normal adaptation schemes. The theoretical predictions, which were verified experimentally, confirmed that the scheme proposed by Bäck (see above) gets "stuck" in suboptimal regions of the search space with a low, or zero, mutation rate attached to each member of the population. The results confirmed that a more robust problem-solving mechanism can simply be achieved by ignoring the first step of the algorithm above, and instead using a fixed learning rate as the probability of applying bitwise mutation to the encoding of the strategy parameters in the second step.

Elsewhere an alternative theoretical approach has been developed by Stephens et al. [79]. Their models expanded on concept of neutrality in mapping, and showed that the optimal mutation rate is not only problem but population dependant. They were able to show theoretically that beneficial adaptation of mutation rates can arise from asymmetry in the genotype to phenotype redundancy.

3.3 Conclusions from Self-Adaptive Mutation

As we have indicated above, there is now a wealth of evidence illustrating that in the continuous domain Self-Adaptation is a useful and effective method for automatically adjusting a vital parameter of the search heuristic. This evidence is backed up by a sound theoretical understanding and analysis.

When applied to discrete encodings, there are similarly impressive experimental results, and some of the findings transfer well. For example, truncation selection appears to offer benefits, although self-adaptation can also be observed even using simple Fitness-Proportionate selection implemented via the roulette-wheel algorithm. However, some of the arguments regarding adaptation to the local topology do not translate so well, since the "locality" is purely defined by the choice of variation operator and its parameters. as Smith points out in [74], the value of truncation selection is that it creates multiple copies of the same point in space. When coupled with the adaptation of the mutation strategy first, this means that multiple strategies are tried out for their value at that point in space, or to see it another way, many different search neighbourhoods are tried out for each parent solution.

Several authors have observed that maintaining a diversity of search strategies is vital in order to permit evolution within the space of different strategies. As Stone and Smith point out, for combinatorial problems the link between mutation probabilities and the effect of the mutation operator is not straightforward, since a Bernouilli process is involved to find out how many bits of the solution encoding are to be changed. Thus an apparent diversity of strategies may in fact all have near-identical effects on the solution encoding. This suggest a reason why the use of a smaller set of more widely spread values, with more frequent and rapid transitions between them, proves advantageous in the discrete scheme. of course this is also made possible by the knowledge that the mutation probabiltiy takes a value between in the range $[0, 0.5]$ for discrete problems, whereas in principle the step size could take any value for a problem in the continuous domain.

4 Self-Adapting Crossover

Not only has self-adaptation been used to adapt mutation parameters within Genetic Algorithms, it has also been used to adapt other operators, such as the choice of recombination operators [77], and but also their definition [57,64,71,73].

Elsewhere, following in analysis of the effects of self-adaptation recombination in continuous domains, Deb and Beyer [14] have proposed that Self-adaptation should have the properties that, children are more likely to be created close to parents, the mean population fitness is unchanged, and that the variance in population fitness should increase exponentially with time on flat landscapes. They (and others) showed that for continuous variables appropriately defined crossover operators such as their SBX operator can demonstrate these properties in what they term implicit self-adaptation. As noted above, for combinatorial problems the concept of "nearness" depends very much on the choice of recombination

or mutation operator, so again some of their arguments do not translate. However, within the field of combinatorial optimisation, there has been much related work on linkage evolution, so we will focus here on three main examples which illustrate the issues and successes of self-adaptive recombination.

4.1 Self-Adapting the Choice of Recombination Operator

In this work self-adaption was used to control the choice of which pre-defined crossover operator should be employed whenever two parents were mated [77]. Spears achieved this by adding a bit to the end of each candidate solution indicating whether one point or uniform crossover would be used, and subjecting this to mutation at the same rate as the rest of the genome. He compared two variants. The "Individual level" model used the crossover encoded by pairs of parents (breaking ties evenly). In contrast the "Population level" model measured the proportion of parents encoding for 1X at the start of each generation and then used this as the probability of using 1x for any pair. Interestingly, although these two are statistically the same (at a coarse level), he observed that better results were obtained with individual level adaptation. Both, incidentally, outperformed the algorithms using fixed crossover probabilities.

4.2 Self-Adapting the Definition of Recombination Operator

There has been much interest over the years in the what has been termed "gene-linkage" – a way of considering how probable it is that combinations of allele values in genes will be transmitted together during recombination. A long sequence of works by Goldberg's group in particular (e.g. [27,32] to name but a few) focused on re-ordering genes so that the linkage bias of one or two point crossover could be more effectively applied focussed. In [64] Smith showed how the behaviour of different operators could be modelled mathematically via the operation of a fairly simple function on a structure he called a "linkage array" which can be considered to be attached to each genome. It was shown that both the N-point and Uniform families of Crossover Operators can be defined in terms of linkage arrays, with the randomness in each instantiation captured by a random vector $\bar{x}$. Similarly other adaptive operators can be defined in terms of linkage arrays, with the random choice of parents and mutation taking the place of $\bar{x}$. Independently, other authors such as Kargupta had come to similar conclusions in the creation of adaptive (but not self-adaptive) algorithms such as the Gene Expression Messy GA [38] and the SEARCH framework for describing heuristic search [39]. Smith's argument was that one way of specifying an adaptive recombination operator, which can adjust its arity and linkage arrays to the problem in hand, is to encode the linkage array within the genome of each individual and to use self adaptation to govern the growth of different linkage sets, and hence recombination strategies. A benefit of using a separate binary linkage array for each individual is that it greatly reduces the size of the search problem compared to using real values (i.e. a probabilistic array). The benefits of evaluating different strategies, which accrue naturally from a probabilistic array in a population level approach, are achieved via the use of multiple (μ) binary arrays.

Punctuated Crossover

In the Punctuated Crossover mechanism [57], self adaptation is used to govern the evolution of the linkage array. The problem representation is augmented by the addition of a binary flag between each pair of adjacent loci, which encodes for a crossover point. Successive genes (and crossover bits) are copied from the first parent into the offspring until a crossover point is encountered in either parent. At this point genes start to be copied from the other parent, *regardless* of whether it coded for crossover at that point. This continues until a full child is created, and a second child is created as its complement. Unfortunately this implementation did not preserve much linkage information from either parent. It was demonstrated in the original paper that it was possible for two parents with "medium" numbers of crossover points to create two offspring, such that the linkage array attached to one child contained all the crossover points from both parents.

Linkage Evolving Genetic Operator (LEGO)

Based on the analysis of the Punctuated Crossover algorithm, and the concept of modeling crossover via a linkage array, *LEGO* [61,71,73] was developed as a means of making explicit the self-adaptation of the linkage array to test the viability of this approach. In brief, the operator works by attaching a linkage array to each member of the population, which partitions the loci in that member into a number of linkage sets. The initial implementation of *LEGO* only permitted linkage to occur between adjacent loci, i.e. the evolved linkage sets consist of chains of linked genes. This has an obvious potential weakness in that it cannot capture tight linkage between two non-adjacent genes unless the entire set of genes between them is also linked i.e. it places a heavy emphasis on the chosen representation, unlike some of the schemes described above. The rationale for this was that as has already been noted that learning linkage is a second order effect, and that there is often a problem with allele convergence preventing the algorithm from finding good "blocks". By only considering adjacent linkage, the size of the problem space is reduced from $O(l^2)$ to $O(l)$. Furthermore, when the practical implementation is considered, problems of conflict resolution are immediately found with the use of non-adjacent linkage, which require an arbitration process via which would be decided the parts of the accrued linkage information to be ignored during recombination.

The operator is not restricted to selecting from only two parents, so the genome can be considered as being comprised of a number of distinct linkage sets, or blocks. Unlike the case for two–parent recombination, alternate blocks on the same parent are not linked. When an offspring is created, blocks are chosen sequentially from left to right. In the first locus a parent (denoted X_1) is chosen according to the action of selection operator on the entire population. This block is then copied whole into the new individual, and provided that the end of the genome has not been reached, there is now a new competition to select the next block. Because the whole purpose of the operator is to preserve

information carried in the linkage array, the selection of partial blocks is forbidden. Thus the selection of the next parent is restricted to those with eligible blocks; i.e. the next block selected must have a left hand edge abutting the right-most edge of the partially completed offspring. This requires a dynamic calculation to determine the action of the selection operator on the restricted set of permissible parents. Having chosen a parent X_2 the corresponding block is copied into the new individual and the process repeats until the offspring is fully specified. Note that it is guaranteed to be possible to complete an offspring since succeeding blocks can be chosen from the same parent.

Using O_i to denote the ith offspring produced and $O_{i,j}$ to denote the j^{th} locus of that offspring, the full definition of the LEGO recombination operator is given by:

$$O_{i,j} = \begin{cases} X_{1,j} & j = 1 \\ X_{k,j} & Linked(X_{k,j-1}, X_{k,j}) 1 \leq j \leq l \\ X_{k+1,j} & \neg Linked(X_{k,j-1}, X_{k,j}) \end{cases} \quad (4)$$

where $Linked()$ denotes whether the corresponding genes are linked according to the parents linkage array, and the parents X_k, X_{k+1} are selected from the population as required (i.e. whenever the linkage criterion is not met) using the modified selection probabilities.

In [61], the effects of iterated recombination and mutation are examined. It is shown that for an infinite population, in the absence of any selection pressure with respect to linkage, the system will rapidly evolve to a steady state. This steady state will retain all four combinations of links between two adjacent genes, and will thus retain the ability to adapt to changing circumstances. The proportion of genes linked at the steady state is entirely independent of the starting conditions, being solely a function of the mutation rate, p, applied.Thus any deviation from this pattern indicates a selection pressure for or against linkage. In practice, analysis of the evolved linkage on a number of different problem types [74] showed that when the "true" linkage pattern was adjacent, then appropriate linkage patterns were observed to form and propagate. In contrast if the linkage was non-adjacent then highly linked chromosomes tended to evolve, i.e. the system evolved to avoid disrupting groups of co-adapted genes, at the expenses of mixing. Note that possible ways of avoiding this problem include re-ordering prior to chain linkage (as is done in the MIMIC algorithm) or removing the restriction to adjacent linkage by proving a means of block conflict resolution.

In [64] a series of experiments were made comparing fixed crossover strategies, 3 different variants of Population level LEGO based on statistics of the linkage arrays(with 3 different variants) and Component level LEGO. These were similar in spirit to those described above performed by Spears. Results show that LEGO outperforms 1-point and uniform crossover on most problems, but that getting the scope right is vital. This is illustrated in Figure 4.2 which shows the success rate and time to success for different algorithms on progressively longer functions made by concatenating Deb's Deceptive function [19]. As can be seen, providing it is properly formulated, self-adaptation permits the rapid identification and

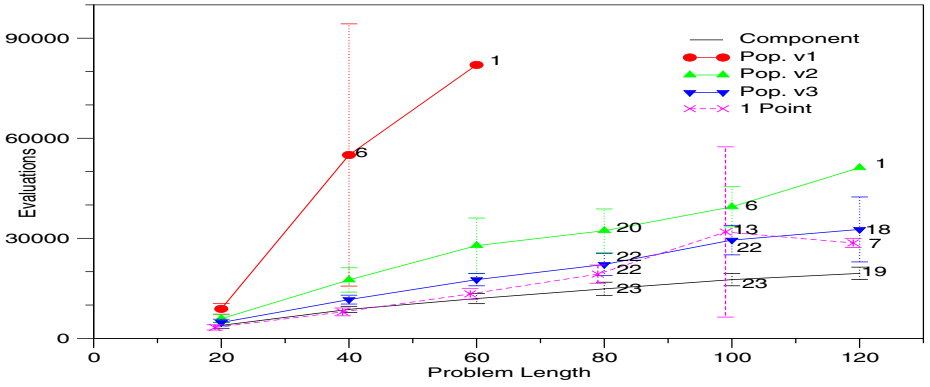


Fig. 5. Mean and Standard Deviations of time taken to solve problems as a function of length. Where not all runs found the global optimum, the number of successful runs (used to calculate statistics) is shown.

mixing of appropriate blocks of genes within the problem encoding. By contrast crossover using a single randomly chosen point fails to reliably solve the problem beyond a moderate size, and uniform crossover failed to solve any of the problems in the time allowed.

Finally we should point out that diversity plays a perhaps more vital role when adapting recombiation (or linkage). Thus for example Tuson and Ross failed to observe any benefits when they tried to use self-adaption to evolve the probability of applying fixed crossover operator [1]. The reason for this is simple: since recombination does not introduce new allele values, but simply shuffles them, once a population has converged in at two adjacent loci, then it makes no difference whether or not crossover is applied between those genes, as the offspring will be identical in the two cases.

5 Self-Adapting Multiple Operators

So far we have considered how self-adaptation may be used to successfully search the space of possible parameter settings or definitions for a single variation operator, while the others are still left for the designer to decide. However, the rationale for research into adaptive algorithms suggests that it is worth considering whether this learning approach can be used to control more than one variation operator, thus further reducing the potential pitfalls of user-set, and static values. The literature contains many examples of adaptive schemes being used to control multiple operators (see e.g. [20,75] for examples) and a few authors have experimented with self-adaptive schemes.

Smith's "Adaptively Parameterised Evolutionary Systems" (APES) algorithm combined the LEGO algorithm described above with self-adaptation of the mutation rates attached to each block of genes [61,72]. Since the blocks are dynamically defined by self-adaption of the linkage array, a separate mutation rate is

self-adapted for each gene, and when genes are linked in a block, their average mutation rate is applied to the block. Comparison results were reported on a range of NK functions with very different characteristics [40] showed that there was a synergy between the two forms of adaptation that significantly outperformed either form of self-adaptation in its own, and dramatically outperformed a algorithms using different combinations of fixed mutation rates and recombination operators.

Elsewhere Hinterding et al. added an adaptive population resizing heuristic and self-adaptation of crossover to that of mutation rates with some success [34]. Mutation, crossover, and population size are all controlled on-the-fly in the GA "without parameters" of Bäck et al. in [9]. This uses self-adaptive mutation from [4], a new self-adaptive technique for adapting crossover rates of the individuals, and Arabas' GAVaPS lifetime idea [3] for population sizing. This study differs from those discussed before in that it explicitly compares GA variants using only one of the (self-)adaptive mechanisms and the GA applying them all. The experiments show remarkable outcomes: the completely (self-)adaptive GA wins, closely followed by the one using only the adaptive population size control, and the GAs with self-adaptive mutation and crossover are significantly worse. These results suggest that putting effort into adapting the population size could be more effective than trying to adjust the variation operators. This is truly surprising considering that traditionally the on-line adjustment of the variation operators has been pursued and the adjustment of the population size received relatively little attention. The subject certainly requires more research.

6 Extension to Memetic Algorithms

6.1 Self-Adapting the Choice of Local Search Operators

So far we have attempted to answer the first three questions that were posed in the introduction. This now leaves the fourth - namely whether the basic ideas of self-adaptation can be taken into other meta-heuristic settings. This remains an open question as far as most other algorithms are concerned. However there have been a number of papers in which the ideas have been incorporated into the related field of Memetic Algorithms (MAs) and used to control the choice, or even definition of the local search operator.

In the "Multi-memetic algorithms" [41,42,44,47] an extra gene was added to each member of the population which encoded the choice of Local Search operator (meme). The mechanism used was based on Smith's discrete self-adaptation scheme for mutation rates described above. Thus local searchers came from a finite set using different move operators and depths of search, and the choice was inherited from the parents, and randomly reset with a small mutation probability. Comparison tests were run with MAs using each of the fixed strategies, and observations of the performance plots showed that "best" meme - i.e. the learning strategy that yielded the best solutions, changed as a function of evolutionary time. The results for the self-adaptive multi-memetic algorithm showed that it was able to track the performance of current best fixed meme, and obtain

better quality solutions than any of the non-adaptive algorithms over a range of problem types such as NK landscapes, TSP, Protein Structure prediction and protein structure comparison. This observation that the choice of "optimal" local search strategy depends on the current state if the population is exactly what has been observed many times before for mutation and recombination, but had not previously been shown within the context of memetic algorithms. Since then it has sparked considerable interest and research activity in what have been called "Adaptive Memetic Algorithms", and a good review of work in that area can be found in [52].

6.2 Self-Adapting the Definition of Local Search Operators

Krasnogor and Gustaffson [43,45,46] suggested an approach based on specifying a grammar for memes describing what local search method should be used, and when in EA cycle. They have proposed that memes could be self-adapted as words in this grammar and shown some promising initial results on bioinformatics problems.

The CO-evolution of Memetic Algorithms (COMA) framework is a general framework for the coevolution of populations of memes and genes in memetic algorithms [63,65,67,68,69,70]. Here Memes are encoded as tuples of $<$ *depth, pivot, pairing, condition, action*$>$, where *Condition* and *action* are patterns to match and replace in the solution encoding. If the pairing takes the value *linked* then memes are inherited, recombined, and mutated with the genes, so the system is effectively self-adapting the choice of local search method to be applied to each member of the population.

The reported results show that using self-adaptation within COMA shows fast scalable optimisation on a range of problems. If there is a problem structure that can be exploited, then the system rapidly adapts the rule length to the problem's structure, and so quickly finds and exploits building blocks. Conversely, if there is no structure to exploit, and the rules (memes) all have similar fitnesses, then there is effectively a flat landscape for self-adaptation to explore, and the population of memes drifts. The effects is that it keeps evolving local search neighbourhoods, which can provide a means of escape from local optimum.

We can illustrate this phenomena of self-adapting rule learning on different problems based on multiple copies of Deb's 4 bit deceptive function. The fitness of each subproblem i is given by its unitation $u(i)$ (i.e. the number of bits set to 1):

$$f(i) = \begin{cases} 0.6 - 0.2u(i) & : \quad u(i) < 4 \\ 1 & : \quad u(i) = 4 \end{cases} \tag{5}$$

In addition to a concatenated version (which we will refer to as 4-Trap), a second "distributed" version (Dist-Trap)was used in which the subproblems were interleaved i.e. sub-problem i was composed of the genes $i, i+16, i+32, i+48$. This separation ensures that even the longest rules allowed in these experiments would be unable to alter more than one element in any of the subfunctions. A third variant of this problem (Shifted-Trap) was designed to be more difficult

than the first for the COMA algorithm to learn a single generalisation, by making patterns which were optimal in one sub-problem, sub-optimal in all others. This was achieved by noting that each sub-problem as defined above is a function of unitation, and therefore can be arbitrarily translated by defining a 4-bit string and using the Hamming distance from this string in place of the unitation. Since we have 16 sub-problems, we simply used the binary coding of the sub-problem's index as basis for its fitness calculation.

A simple Ga was compared to a simple MA - i.e. one which applied a bit-flipping hill-climber to each member of the population. Also tested were versions of COMA with fixed rule lengths, and a version in which the rule length was encoded and subjected to self-adaptation. All features such as population size, mutation probability, crossover, selection/replacement strategies and the total number of calls to the evaluation function permitted to the algorithm were the same in every case.

Figure 6 shows the results of these experiments as a plot of mean time to optimum for 4-Trap with three different population sizes. When an algorithm failed to reach the optimum in all twenty runs, the mean is taken over the successful runs, and this number is shown. The error bars represent one standard deviation. It should be noted that the scale on the y-axis is logarithmic. We can see that the GA and MA,and 1-Coma algorithms fail to find the optimum as frequently, or when they do as fast, for the smaller population sizes. For all population sizes there is greater variance in the performance of these three algorithms than for the other variants. Statitistical analysis confirmed these results. Note that the 1-COMA algorithm is faster and more reliable than the simple MA. Inspection shows that this is because it self-adaptation preserves the rule $0 \rightarrow 1$ but discards the rule $1 \rightarrow 0$, whereas of course the simple bit-flipping MA uses both.

In short, what we can observe is that for fixed rule lengths of between 3 and 9, and for the adaptive version, the COMA system derives performance benefits from evolving LS rules. Significantly, and unlike the GA and MA, the COMA algorithm does not depend on a certain size population before it is able to solve the problem reliably. This is indicative of a far more scaleable algorithm.

In order to examine the behaviour of the algorithm the population mean values were observed for the effective rule length (only relevant for A-Coma), the "specificity" (i.e. the proportion of values in the condition not set to #) and the "unitation" (the proportion of bits in the action set to 1), and also the highest fitness in the population (with 100 as the optimum) as a function of the number of elapsed generations. Figure 7 shows the A-Coma results averaged over 20 runs on each of the three problems, with a population of 250. The evolving rule bases were manually inspected on a large number of runs for each problem.

For the 4-Trap function (left hand graph), the system rapidly evolves medium length $(3-4)$, general (specificity $< 50\%$) rules whose action is to set all the bits to 1 (mean unitation approaches 100%). Note that in the absence of selective pressure (i.e. the pivot rules meant that the solutions were left unchanged), all three of these values would be expected to remain at their initial values, so these

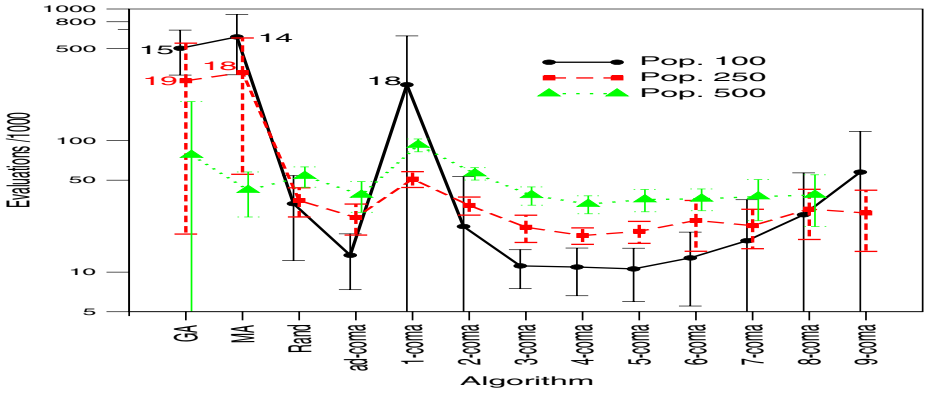


Fig. 6. Times to optimum for the 4-Trap function. Note logarithmic y-axis.

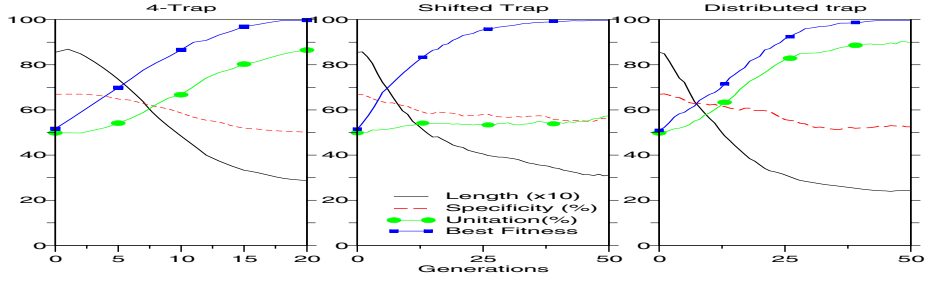


Fig. 7. Analysis of Evolving Rules by Function Type

changes result from beneficial adaptation. Closer inspection of the evolving rule base confirms that the optimal subproblem string is being learned and applied.

For the Shifted-Trap function, where the optimal sub-blocks are all different (middle) the rule length decreases more slowly. The specificity also remains higher, and the unitation remains at 50%, indicating that different rules are being maintained. This is borne out by closer examination of the rule sets.

The behaviour on Dist-Trap is similar to that on 4-Trap, albeit over a longer timescale. Rather than learning specific rules about sub-problems, which cannot possibly be happening (since no rule is able to affect more than one locus of any subproblem), the system is apparently learning the general rule of setting all bits to 1. The rules are generally shorter than for 4-Trap, (although this is slightly obscured by the averaging) which means that the number of potential neighbours is higher for any given rule. Equally, the use of wildcard characters, coupled with the fact that there may be matches in the two parts of the rules, means that length of the rules used defines a maximum radius in Hamming space for the neighbourhood, rather than a fixed distance from the original solution. Both of these observations, when taken in tandem with the longer times to solution,

suggest that when the system is unable to find a single rule that matches the problems' structure, a more diverse search using a more complex neighbourhood is used, which slowly adapts itself to the state of the current population of solutions.

These results shows how self-adaptation is able to learn not just which if a fixed self of local search methods to apply to each individual, but also to adapt the definition of the local search strategy. Subsequent published results have confirmed the benefits of this approach across a spread of problem types.

7 Summary and Conclusions

The basic message of this paper is that Self-adaptation works, for a range of different representations and for a range of different operators, and that it need not be viewed simply as a means of controlling mutation rates. Hopefully it has also become apparent that a diversity of strategies is a vital component if evolution is to be used as a learning mechanism within the space of possible search strategies, and that this has especial implications for combinatorial problems.

In order for self-adaptation to work as a learning mechanism for controlling search strategies the basic requirement is that a diverse set of strategies is compared and used according to their value when applied to the *current* set of solutions. Diversity comes from the action of the search operators on an encoding of the search strategy, and is maintained by the update function. In an EA, these are mutation and the selection operator respectively, but the range of different approaches highlighted above suggests that this need not be the case. This suggests that the model could be applied to other population based algorithms such as Ant Colony Optimisation, Learning Classifier Systems and parallel versions of Simulated Annealing or Tabu Search. These remain as promising areas for future research.

Acknowledgement. Jim Smith would like to thank Terry Fogarty, Chris Stone, Gusz Eiben and Nat Krasnogor for many interesting discussions in this area.

References

1. Adapting operator settings in genetic algorithms. Evolutionary Computation 6(2), 161–184 (1998)
2. Angeline, P.J.: Adaptive and self-adaptive evolutionary computations. In: Computational Intelligence, pp. 152–161. IEEE Press, Los Alamitos (1995)
3. Arabas, J., Michalewicz, Z., Mulawka, J.: Gavaps - a genetic algorithm with varying population size, pp. 73–78 (1994)
4. Bäck, T.: The interaction of mutation rate, selection and self-adaptation within a genetic algorithm. In: Männer, R., Manderick, B. (eds.) Proceedings of the 2nd Conference on Parallel Problem Solving from Nature, pp. 85–94. North-Holland, Amsterdam (1992)
5. Bäck, T.: Self adaptation in genetic algorithms. In: Varela and Bourgine [82], pp. 263–271

6. Bäck, T.: Optimal mutation rates in genetic search. In: Forrest [23], pp. 2–8
7. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford (1996)
8. Bäck, T.: Self-adaptation. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) Evolutionary Computation 2: Advanced Algorithms and Operators, ch. 21, pp. 188–211. Institute of Physics Publishing, Bristol (2000)
9. Bäck, T., Eiben, A.E., van der Vaart, N.A.L.: Proceedings of the 6th Conference on Parallel Problem Solving from Nature. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 315–324. Springer, Heidelberg (2000)
10. Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Institute of Physics Publishing, Bristol, and Oxford University Press, New York (1997)
11. Bäck, T., Hofmeister, F., Schwefel, H.P.: A survey of evolution strategies. In: Belew and Booker [13], pp. 2–9
12. Bäck, T., Schütz, M.: Intelligent mutation rate control in canonical genetic algorithms. In: Ras, Z. (ed.) Proceedings of the Ninth International Symposium on Methodologies for Intelligent Systems, pp. 158–167. Springer, Heidelberg (1996)
13. Belew, R.K., Booker, L.B. (eds.): Proceedings of the 4th International Conference on Genetic Algorithms. Morgan Kaufmann, San Francisco (1991)
14. Beyer, H., Deb, K.: On self-adaptive features in real-parameter evolutionary algorithms. IEEE Transactions on Evolutionary Computation 5(3), 250–270 (2001)
15. Beyer, H.-G.: The Theory of Evolution Strategies. Springer, Berlin (2001)
16. 2003 Congress on Evolutionary Computation (CEC 2003). IEEE Press, Piscataway (2003)
17. Davis, L.: Adapting operator probabilities in genetic algorithms. In: Schaffer [54], pp. 61–69
18. De Jong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)
19. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) Foundations of Genetic Algorithms 2, pp. 93–108. Morgan Kaufmann, San Francisco (1992)
20. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (1999)
21. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computation. Springer, Heidelberg (2003)
22. Fogel, D.B.: Evolutionary Computation. IEEE Press, Los Alamitos (1995)
23. Forrest, S. (ed.): Proceedings of the 5th International Conference on Genetic Algorithms. Morgan Kaufmann, San Francisco (1993)
24. Glickman, M., Sycara, K.: Evolutionary algorithms: Exploring the dynamics of self-adaptation, pp. 762–769 (1998)
25. Glickman, M., Sycara, K.: Reasons for premature convergence of self-adaptating mutation rates. In: 2000 Congress on Evolutionary Computation (CEC 2000), pp. 62–69. IEEE Press, Piscataway (2000)
26. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
27. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3(5), 493–530 (1989)
28. Grefenstette, J.J.: Optimisation of control parameters for genetic algorithms. IEEE Transaction on Systems, Man and Cybernetics 16(1), 122–128 (1986)

29. Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439. Springer, Heidelberg (2002)
30. Hansen, N.: An analysis of mutative σ-self-adaptation on linear fitness functions. Evolutionary Computation 14(3), 255–275 (2006)
31. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
32. Harik, G., Goldberg, D.E.: Learning linkage. Technical Report IlliGAL 96006, Illinois Genetic Algorithms Laboratory, University of Illinois (1996)
33. Hesser, J., Manner, R.: Towards an optimal mutation probablity in genetic algorithms. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 23–32. Springer, Heidelberg (1991)
34. Hinterding, R., Michalewicz, Z., Peachey, T.C.: Self adaptive genetic algorithm for numeric functions. In: Voigt et al. [83], pp. 420–429
35. Proceedings of the 1996 IEEE Conference on Evolutionary Computation. IEEE Press, Piscataway (1996)
36. Jain, A., Fogel, D.B.: Case studies in applying fitness distributions in evolutionary algorithms. II. Comparing the improvements from crossover and Gaussian mutation on simple neural networks. In: Yao, X., Fogel, D.B. (eds.) Proc. of the 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, pp. 91–97 (2000)
37. Julstrom, B.A.: What have you done for me lately?: Adapting operator probabilities in a steady-state genetic algorithm. In: Eshelman, L.J. (ed.) Proceedings of the 6th International Conference on Genetic Algorithms, pp. 81–87. Morgan Kaufmann, San Francisco (1995)
38. Kargupta, H.: The gene expression messy genetic algorithm. In: ICEC-96 [35], pp. 814–819
39. Kargupta, H., Bandyopadhyay, S.: A perspective on the foundation and evolution of the linkage learning genetic algorithms. J Computer Methods in Applied Mechanics and Engineering 2186, 266–294 (2000)
40. Kauffman, S.A.: Origins of Order: Self-organization and Selection in Evolution. Oxford University Press, New York (1993)
41. Krasnogor, N.: Coevolution of genes and memes in memetic algorithms. In: Wu, A.S. (ed.) Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program (1999)
42. Krasnogor, N.: Studies in the Theory and Design Space of Memetic Algorithms. PhD thesis, University of the West of England (2002)
43. Krasnogor, N.: Self-generating metaheuristics in bioinformatics: The protein structure comparison case. Genetic Programming and Evolvable Machines 5(2), 181–201 (2004)
44. Krasnogor, N., Blackburne, B.P., Burke, E.K., Hirst, J.D.: Multimeme algorithms for protein structure prediction. In: Guervos et al. [29], pp. 769–778
45. Krasnogor, N., Gustafson, S.: Toward truly "memetic" memetic algorithms: discussion and proofs of concept. In: Corne, D., Fogel, G., Hart, W., Knowles, J., Krasnogor, N., Roy, R., Smith, J., Tiwari, A. (eds.) Advances in Nature-Inspired Computation: The PPSN VII Workshops, pp. 9–10. University of Reading, Reading, UK (2002); PEDAL (Parallel, Emergent & Distributed Architectures Lab)
46. Krasnogor, N., Gustafson, S.M.: A study on the use of "self-generation" in memetic algorithms. Natural Computing 3(1), 53–76 (2004)
47. Krasnogor, N., Smith, J.E.: Emergence of profitable search strategies based on a simple inheritance mechanism. In: Spector et al. [78], pp. 432–439

48. Lee, M., Takagi, H.: Dynamic control of genetic algorithms using fuzzy logic techniques. In: Forrest [23], pp. 76–83
49. Liang, K.-H., Xao, X., Liu, Y., Newton, C., Hoffman, D.: An experimental investigation of self-adaptation in evolutionary programming (1998)
50. Lis, J.: Parallel genetic algorithm with dynamic control parameter. In: ICEC-96 [35], pp. 324–329
51. Meyer-Nieberg, S., Beyer, H.G.: Self-adaptation in evolutionary algorithms. In: Parameter Setting in Evolutionary Algorithms, pp. 47–75 (2007)
52. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: A comparative study. IEEE Transactions on Systems Man and Cybernetics Part B 36(1) (2006)
53. Rudolph, G.: Self-adaptive mutations lead to premature convergence. IEEE Transactions on Evolutionary Computation 5, 410–414 (2001)
54. Schaffer, J.D. (ed.): Proceedings of the 3rd International Conference on Genetic Algorithms. Morgan Kaufmann, San Francisco (1989)
55. Schaffer, J.D., Caruana, R.A., Eshelman, L.J., Das, R.: A study of control parameters affecting online performance of genetic algorithms for function optimisation. In: Schaffer [54], pp. 51–60
56. Schaffer, J.D., Eshelman, L.J.: On crossover as an evolutionarily viable strategy. In: Belew Booker [13], pp. 61–68
57. Schaffer, J.D., Morishima, A.: An adaptive crossover distribution mechanism for genetic algorithms. In: Grefenstette, J.J. (ed.) Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 36–40. Lawrence Erlbaum, Hillsdale (1987)
58. Schlierkamp-Voosen, D., Mühlenbein, H.: Strategy adaptation by competing subpopulations. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 199–209. Springer, Heidelberg (1994)
59. Schwefel, H.-P.: Numerische Optimierung von Computer-Modellen Mittels der Evolutionsstrategie. ISR, vol. 26. Birkhaeuser, Basel/Stuttgart (1977)
60. Schwefel, H.-P.: Numerical Optimisation of Computer Models. Wiley, New York (1981)
61. Smith, J.E.: Self Adaptation in Evolutionary Algorithms. PhD thesis, University of the West of England, Bristol, UK (1998)
62. Smith, J.E.: Modelling GAs with self-adaptive mutation rates. In: Spector et al. [78], pp. 599–606
63. Smith, J.E.: Co-evolution of memetic algorithms: Initial investigations. In: Guervos et al. [29], pp. 537–548
64. Smith, J.E.: On appropriate adaptation levels for the learning of gene linkage. J. Genetic Programming and Evolvable Machines 3(2), 129–155 (2002)
65. Smith, J.E.: Co-evolving memetic algorithms: A learning approach to robust scalable optimisation. In: CEC 2003, [16], pp. 498–505 (2003)
66. Smith, J.E.: Parameter perturbation mechanisms in binary coded gas with self-adaptive mutation. In: Rowe, P., De Jong, Cotta (eds.) Foundations of Genetic Algorithms 7, pp. 329–346. Morgan Kaufmann, San Francisco (2003)
67. Smith, J.E.: Protein structure prediction with co-evolving memetic algorithms. In: CEC 2003 [16], pp. 2346–2353 (2003)
68. Smith, J.E.: The co-evolution of memetic algorithms for protein structure prediction. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms, pp. 105–128. Springer, New York (2004)
69. Smith, J.E.: Co-evolving memetic algorithms: A review and progress report. IEEE Transactions in Systems, Man and Cybernetics, part B 37(1), 6–17 (2007)

70. Smith, J.E.: Credit assignment in adaptive memetic algorithms. In: Proceedings of Gecco, the ACM-SIGEVO conference on Evolutionary computation, pp. 1412–1419 (2007)
71. Smith, J.E., Fogarty, T.C.: An adaptive poly-parental recombination strategy. In: Fogarty, T.C. (ed.) Evolutionary Computing 2, pp. 48–61. Springer, Berlin (1995)
72. Smith, J.E., Fogarty, T.C.: Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm. In: Voigt et al. [83], pp. 441–450
73. Smith, J.E., Fogarty, T.C.: Recombination strategy adaptation via evolution of gene linkage. In: ICEC-96 [35], pp. 826–831 (1996)
74. Smith, J.E., Fogarty, T.C.: Self adaptation of mutation rates in a steady state genetic algorithm. In: ICEC-96 [35], pp. 318–323 (1996)
75. Smith, J.E., Fogarty, T.C.: Operator and parameter adaptation in genetic algorithms. Soft Computing 1(2), 81–87 (1997)
76. Smith, R.E., Smuda, E.: Adaptively resizing populations: Algorithm, analysis and first results. Complex Systems 9(1), 47–72 (1995)
77. Spears, W.M.: Adapting crossover in evolutionary algorithms. In: McDonnell, J.R., Reynolds, R.G., Fogel, D.B. (eds.) Proceedings of the 4th Annual Conference on Evolutionary Programming, pp. 367–384. MIT Press, Cambridge (1995)
78. Spector, L., Goodman, E., Wu, A., Langdon, W.B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., Burke, E. (eds.): Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001). Morgan Kaufmann, San Francisco (2001)
79. Stephens, C.R., Garcia Olmedo, I., Moro Vargas, J., Waelbroeck, H.: Self-adaptation in evolving systems. Artificial Life 4, 183–201 (1998)
80. Stone, C., Smith, J.E.: Strategy parameter variety in self-adaption. In: Langdon, W.B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M.A., Schultz, A.C., Miller, J.F., Burke, E., Jonoska, N. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), July 9–13, 2002, pp. 586–593. Morgan Kaufmann, San Francisco (2002)
81. Syswerda, G.: A study of reproduction in generational and steady state genetic algorithms. In: Rawlins, G. (ed.) Foundations of Genetic Algorithms, pp. 94–101. Morgan Kaufmann, San Francisco (1991)
82. Varela, F.J., Bourgine, P. (eds.): Toward a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life. MIT Press, Cambridge (1992)
83. Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.): Proceedings of the 4th Conference on Parallel Problem Solving from Nature. PPSN 1996. LNCS, vol. 1141. Springer, Heidelberg (1996)
84. Wolpert, D.H., Macready, W.G.: No Free Lunch theorems for optimisation. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)

Part II
New Techniques and Applications

An Efficient Hyperheuristic for Strip-Packing Problems*

Ignacio Araya[1], Bertrand Neveu[1], and María-Cristina Riff[2]

[1] Project COPRIN, INRIA, Sophia-Antipolis, France
{Ignacio.Araya,Bertrand.Neveu}@sophia.inria.fr
[2] Department of Computer Science, Universidad Técnica Federico Santa María, Valparaíso, Chile
Maria-Cristina.Riff@inf.utfsm.cl

Summary. In this paper we introduce a hyperheuristic to solve hard strip packing problems. The hyperheuristic manages a sequence of greedy low-level heuristics, each element of the sequence placing a given number of objects. A low-level solution is built by placing the objects following the sequence of low-level heuristics. The hyperheuristic performs a hill-climbing algorithm on this sequence by testing different moves (adding, removing, replacing a low-level heuristic). The results we obtained are very encouraging and improve the results from the single heuristics tests. Thus, we conclude that the collaboration among heuristics is an interesting approach to solve hard strip packing problems.

Keywords: Hyperheuristic, strip packing problems, low-level heuristic, hill climbing.

1 Introduction

In this paper we focus our attention on methods to solve the two-dimensional strip packing problem, where a set of rectangles (objects) must be positioned on a container (a rectangular space area). This container has a fixed width dimension and a variable height size. The goal is, when possible, to introduce all the objects in the container without overlapping, using a minimum height dimension of the container. This problem is NP-hard and exact approaches [18, 15] are in general limited to small instances. Four variants of this problem exist, depending on the possibility of rotation of the objects, and on the presence of the guillotine cut constraint[1].

In the literature many heuristic approaches have been proposed. In our understanding the most complete review has been presented in E. Hopper's Thesis [11]. However, in the last few years the interest in this subject has increased, and so has the interest in the number of research papers presenting new approaches and improvements to the existing strategies. These approaches are in general single

* This work was partially financed by the Fondecyt Project 1060377.

[1] This constraint requires that all objects placed in the container can be reproduced by a series of guillotine cuts, i.e. edge-to-edge cuts parallel to the edges of the container.

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 61–76, 2008.
springerlink.com

heuristics or heuristics incorporated into metaheuristics methods. Recently, the concept of hyperheuristic has been introduced and successfully tested in different problems, [5]. The key idea is to tackle problems using various low-level heuristics and develop a framework that controls the applications of the heuristics. Using this framework the time consuming task of designing an algorithm with special components for a specific algorithm is reduced. This kind of approach is useful to obtain a good solution for a problem in a reasonable amount of time. It emphasizes a compromise between the quality of the solution and the invested time for designing the algorithm. Our goal is to show that hyperheuristics can be applied to solve Strip Packing Problems providing effective solutions in an efficient way. Our hyperheuristic is compared to other approaches using well known benchmarks. This paper is organized as follows: First we present an overview of methods based on heuristics to solve the strip packing problem, which are included in our hyperheuristic approach. Next we introduce our hyperheuristic. We will then present the results obtained using the benchmarks. Finally, our conclusions and future trends in this research area are presented.

2 Heuristics Based Methods

In this section, we present a survey of the main heuristics for strip packing problems and of the most efficient algorithms using them.

2.1 Various Low-Level Heuristics

Baker in [2] introduced Bottom-Left heuristics (BL), which first orders the objects according to their area. The objects are placed at the top and pushed down and left as much as possible. This method was improved by Chazelle [8] and called Bottom Left Fit (BLF) : each object is located at the most bottom and left possible place. Hopper [12] presented BLD which is an improved version of BL, where the objects are ordered using various criteria (height, width, perimeter, area) and the algorithm selects the best result obtained. Lesh et al. in [16] focus their research on improving BLD heuristic. They call their new heuristics BLD^*. In BLD^* the objects are randomly ordered according to the Kendall-tau distance from all of the possible fixed orders. This strategy is called Bubble Search, [17] and can be applied to any constructive algorithm in order to randomize a fixed ordering. As in GRASP, this strategy repeats greedy placements with this randomized ordering until a time limit is reached.

Another type of heuristics, Best Fit (BF) [6], uses a dynamic ordering for the rectangles to be located. The algorithm goes through the possible places from the most bottom left one, and selects for each place the rectangle that best fits in it (if it exists).

Let us now describe heuristics for problems with guillotine cut constraint. The heuristics FFDH and NFDH proposed in [14] and BFDH proposed initially in [19], and modified by [3] as BFDH* are very similar. In each of them, the objects are oriented such that their width is not lower than their height, and

they are ordered from highest to lowest. Each object is packed in a rectangular sub-area of the container in the bottom left corner. The width of the sub-area is given by the container, and the height is given by the first object packed in this sub-area. When it is possible to include the current object to be placed into some sub-areas, it is positioned into the sub-area having: the least available area for BFDH; the bottom available area for FFDH; and the top area, if it is available, for NFDH. In other cases the algorithm opens a new sub-area above the existing sub-areas positioning the current object in the bottom left corner as the first object of this sub-area. BFDH* seeks to improve this heuristic by allowing object rotations, so that when the algorithm searches to include the current object into a sub-area it tests both orientations.

Zhang et al. [21] propose a recursive heuristic HR for problems with guillotine cut constraint. When the first object is positioned in the container (on the bottom left corner) it identifies two remaining areas. It recursively continues placing the remaining objects. To improve the performance of the heuristic, the authors present a deterministic algorithm (HRalg) that gives priority to the objects with bigger areas. Zhang et al. claim that their algorithm quickly obtains good results on Hoper's benchmarks.

For our approach we have selected HR, BF, BLF, BFDH* as the low-level heuristics for problems without guillotine cut constraint, because they have shown to be individually competitive. For problems with guillotine cut constraint the selected heuristic are HR and BFDH*.

2.2 Metaheuristic Approaches

These and other low-level heuristics have been used in metaheuristic approaches, as tabu search, simulated annealing, and genetic algorithms. The first idea is to build an initial solution by a low level heuristic and to perform a local search on the layout. Neveu et al. [20] present an incremental move, which allows additions and removals of rectangles. They also implement a generic metaheuristic using this move obtaining competitive results.

Other researchers prefer to work on the order of the objects for each positioning heuristic. In [12] they present a genetic algorithm and a simulated annealing algorithm (GA+BLF and SA+BLF), both of which try to find the best order for the objects to be placed in the container using the BLF strategy.

For the case of fixed orientation problems, the best approach to our knowledge appears to be the GRASP based approach described in [1]. This approach repeats the following two-phases algorithm: the rectangles are first placed by a slightly randomized BF like constructive phase. Then the solution is improved by a strictly improving Variable Neighborhood Search (VNS).

On the other hand, Bortfeldt [3] introduced a Genetic Algorithm called SP-GAL and obtained the best results known in the literature for the problems allowing the rectangles to be rotated. The algorithm generates an initial population using a BFDH* heuristic which is an improvement of the BFDH heuristic initially proposed in [19]. This heuristic works with a layer structure, that takes into account the guillotine cut constraint. The genetic algorithm directly

performs a search in this layer structure. For problems without the guillotine cut constraint, a post-optimization procedure breaks this layer structure. The same genetic algorithm is used in [4] for bigger instances (1000 pieces). It is divided in GA-1, GA-2, GA-3 and GA-4, each of them initialized with different parameters. The procedure is only applied to problems with the guillotine cut constraint, because the post-optimization procedure is negligible for large instances [4].

Burke et al. [7] hybridize the best-fit heuristic with metaheuristic approaches such as tabu search (BF-TS), simulated annealing (BF-SA) and genetic algorithms (BF-GA). BF-SA obtains the best results.

3 The Hyperheuristic Approach: H-SP

The hyperheuristic framework manages a set of low-level heuristic and tries to find a way to apply them. There are some genetic inspired hyperheuristics in the literature to solve combinatorial problems [9,10]. However, in most of the cases, they use a representation that just corresponds to a simple sequence of low-level heuristics to be applied.

We have chosen to build a simple hyperheuristic that manages a sequence of low-level greedy heuristics. From the analysis of the four selected low-level heuristics we can remark the following:

- Performance changes according to the order of the objects and their rotation.
- The data structure to obtain a good implementation code is not always the same for all of these heuristics.

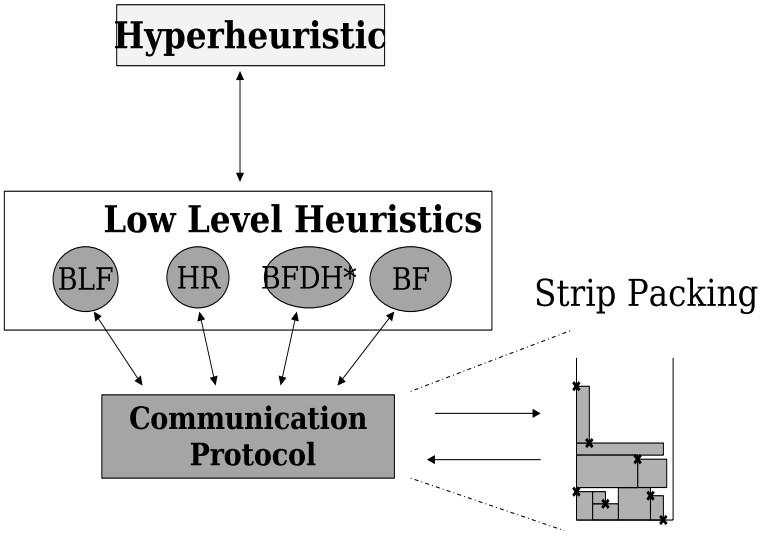


Fig. 1. H-SP: Hyperheuristic for Strip Packing

Taking into account these remarks, we have designed a new hyperheuristic approach which allows us to include a good individual implementation for each heuristic considering them as black boxes. They communicate following a protocol for modifying the current state of the search (the floor with the objects already located by the preceding heuristics and the remaining objects to locate) as shown in figure 1.

3.1 Representation

The representation structure used is a *constructive algorithm* formed by the sequential composition of constructive heuristics among a set H. A **configuration** X is thus a constructive algorithm:

$$X = h_1(p_1, n_1) * h_2(p_2, n_2) * ... * h_k(p_k, n_k) \tag{1}$$

Where $h_1, ..., h_k \in H$ are the constructive heuristics, $p_1, ..., p_k \in P$ are parameters to initialize the heuristics and n_i is an integer number that represents the amount of pieces that the heuristic h_i must place. $*$ is the sequential composition operator.

The sets P and H depend on the kind of problem that will be solved (with or without guillotine constraint, with or without rotation allowed).

Let N be the number of pieces to place inside the container. The next two constraints must be satisfied:

$$n_i > 0, \forall i = 1...k \tag{2}$$

$$\sum_{i=0}^{k} n_i = N \tag{3}$$

The parameters p_i are related to the order and the rotation of the pieces before the placement. The basic order criteria used are: decreasing heights (DP), decreasing widths (DW), decreasing areas (DA) and decreasing perimeters (DP). The rotation criteria used are: width greater or equal than the heights ($W \geq H$), heights greater or equal than the widths ($H \geq W$), rotate no object (NR) and rotate all the objects (All_R).

Figure 2 shows a configuration example with 3 heuristics. To translate the configuration into the problem, the heuristics are evaluated sequentially. The first is BLF, the parameters p indicate that the rectangles must be ordered by decreasing weights (DW) and rotated with their widths greater or equal than their heights ($W \geq H$). Just when the process of ordering and rotation has been realized, the BLF heuristic will begin to place the pieces inside the container ($n = 4$ pieces, corresponding to the white rectangles). The rectangle numbers indicate the placement order of the pieces.

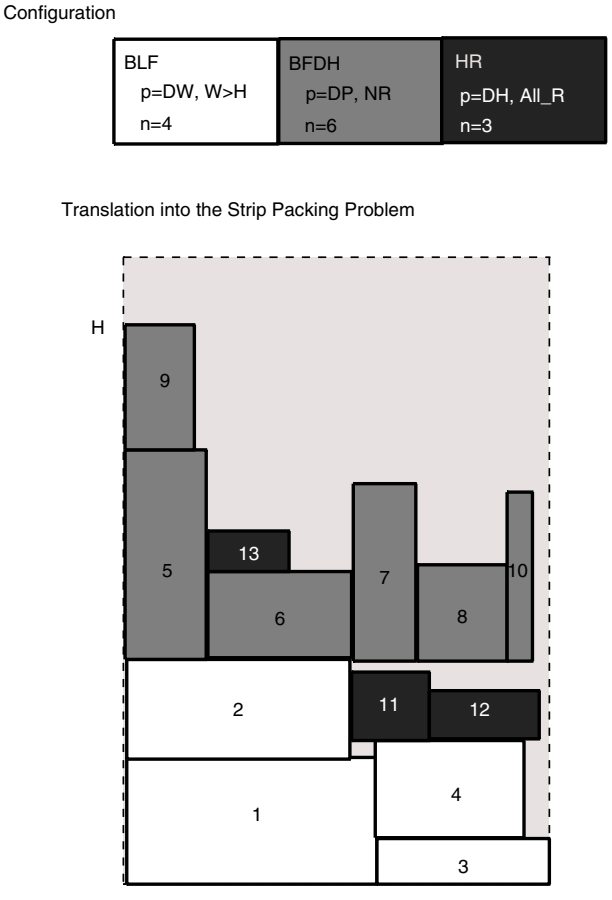


Fig. 2. Configuration example

3.2 Moves

The local search operations that we have defined in our high-level structure allow heuristics to be added, deleted and replaced from the configuration. These operations are applied with equal probability.

Let the current configuration:

$$X_C = h_1(..) * ... * h_{i-1}(p_{i-1}, n_{i-1}) * h_i(p_i, n_i) * h_{i+1}(p_{i+1}, n_{i+1}) * ... * h_k(..) \quad (4)$$

The **add operation** selects random values $i \in \{1..k\}$, $h_{add} \in H$, $p_{add} \in P$ and $n_{add} \in \{1..n_i\}$. The return of the operation is a new configuration:

$$X'_C = ... * h_{i-1}(..) * h_{add}(p_{add}, n_{add}) * h_i(p_i, n_i - n_{add})... \quad (5)$$

If $n_i - n_{add}$ is equal to 0, the heuristic h_i is simply eliminated from the configuration. The key idea of this operation is to include new heuristics in a different step of the algorithm in order to obtain a better cooperation among them.

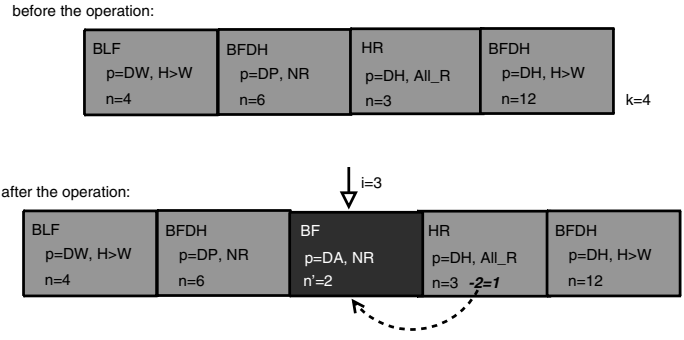


Fig. 3. Example of the *add operation*

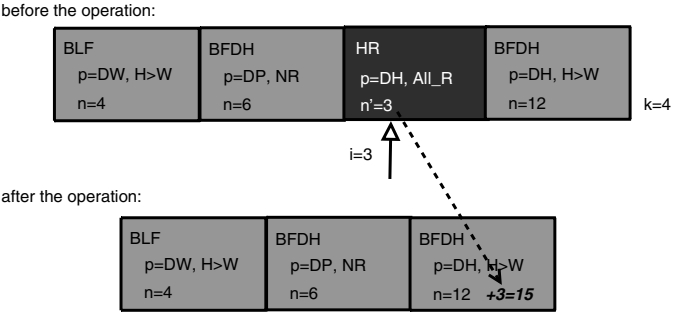


Fig. 4. Example of the *remove operation*

Figure 3 shows an example. The new heuristic is located in the third position of the configuration, reducing by $n'(2)$ the next heuristic n value.

The **remove operation** selects a random value $i \in \{1..k\}$. The return operation is a new configuration:

$$X'_C = ... * h_{i-1}(p_{i-1}, n_{i-1}) * h_{i+1}(p_{i+1}, n_{i+1} + n_i) * ... \tag{6}$$

If the random value of i is equal to k, then:

$$X'_C = ... * h_{k-1}(p_{k-1}, n_{k-1} + n_k) \tag{7}$$

The idea here is to allow the algorithm to discard some heuristics obtaining better results without them.

Figure 4 shows an example. The third heuristic is removed from the configuration and the value of $n'(3)$ is added to the next heuristic n value.

The **replace operation** selects random values $i \in \{1..k\}$, $h_{rep} \in H$ and $p_{rep} \in P$. The operation returned is a new configuration:

$$X'_C = ... * h_{i-1}(..) * h_{rep}(p_{rep}, n_i) * h_{i+1}(..) * ... \tag{8}$$

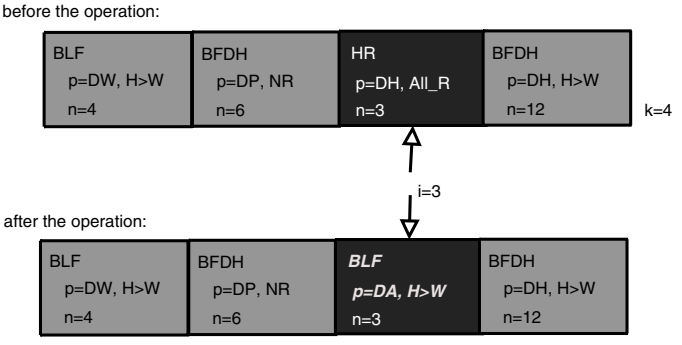


Fig. 5. Example of the *replace operation*

The idea of this operation is to give more exploration capability to the algorithm.

Figure 5 shows an example. The third heuristic in the configuration is replaced by a new one, with new parameters p and the same value of n.

All these operations maintain the constraints (2) and (3) satisfied. The representation and the defined operations allow the hyperheuristic algorithm to reach a wider combination between low-level heuristics.

3.3 Evaluation Function

Our approach uses the traditional fitness function for strip-packing [12], which is to minimize the container's height used. It is supposed that the container's width is fixed. The quality of a constructive algorithm or configuration is evaluated according to the quality of the solution that it obtains.

3.4 Procedure

The hyperheuristic explores the space of constructive algorithms (Xs) by starting from an initial and random generated configuration (X_0). To do that, our approach follows a Hill-climbing procedure, thus in each iteration it is applied one random operation to the algorithm and if the new algorithm X'_C is better or equal than the current one (X_C), then X'_C will be the new algorithm for the next iteration.

In order to escape local minima, we have performed for each H-SP test 10 restarts. It means that one execution of H-SP of 100 seconds corresponds to 10 hill-climbing procedures of 10s each.

The initial algorithm is $X_0 = h_1(p_1, n_1) * h_2(p_2, n_2) * ... * h_m(p_m, n_m)$, with $h_1 \neq h_2 \neq ... \neq h_m$ and $m = \#H$, in other words, *all* the heuristics are used once to construct X_0. The values of p_i are selected at random from the set P, and the values of n_i are fixed satisfying the equation (9):

$$n_i = \frac{i \times N}{m} - \sum_{j=1}^{i-1} n_j \tag{9}$$

Algorithm 1. H-SP($Time\_Limit$)

```
for i = 1 to 10 do
  restart_time()
  X_0 ← RandomAlgorithm(H, P, N)
  Best_Algorithm ← X_0
  X_C ← X_0

  while time()< Time_Limit/10 do
    select RandomNumberFrom(1..3)
      case 1: X'_C ← Add(X_C)
      case 2: X'_C ← Remove(X_C)
      case 3: X'_C ← Replace(X_C)
    end select

    if Evaluate(X'_C) ≥ Evaluate(X_C) then
      X_C ← X'_C
    end if
  end while

  if Evaluate(X_C) ≥ Evaluate(Best_Algorithm) then
    Best_Algorithm = X_C
  end if
  return Best_Algorithm
end for
```

For example, if $m = 4$ ($\#H$ is also 4) and the amount of pieces N is 47, the four heuristics in set H will be selected in some order, the parameters p_i will be randomly selected from P and the values of n_1, n_2, n_3 and n_4 will be respectively 11, 12, 12 and 12.

Algorithm 1 shows the procedure. *RandomAlgorithm* function generates the initial constructive algorithm. *Add*, *Remove* and *Replace* functions, perform the operations described in 3.2. *Evaluate* function executes the generated algorithms and obtains their fitness. Finally the best solution can be obtained executing the *Best_Algorithm*.

4 Tests

We have performed two kinds of tests. The first one compares the results obtained using low-level heuristics with the results of our hyperheuristic approach. We report the quality of the solution found and the percentage used of each single low-level heuristic in the hyperheuristic. The second test compares H-SP with the best reported results from the strip-packing state of the art.

4.1 Benchmarks

For these tests we use the 21 Hopper's instances classified in 7 classes $C_1, \ldots, C_7$, according to their size. The optimal solution of each instance is known, [12]. We

Table 1. Gap to the solution for low-level heuristics and H-SP

Class	**Low-level heuristics**				**H-SP**20s	
	BLF	HR	BFDH*	BF	Average	Best
C1	6.6	6.6	6.6	5	**0**	**0**
C2	13.3	8.8	8.8	8.8	0.89	**0**
C3	11.1	6.6	6.6	6.6	**2.22**	**2.22**
C4	4.4	3.8	3.8	3.3	**1.67**	**1.67**
C5	2.6	2.6	2.6	2.6	1.26	**1.11**
C6	3.1	2.7	2.7	2.5	1.28	**0.83**
C7	2.6	2.6	2.6	2.2	1.17	**0.97**
Average	6.24	4.81	4.81	4.42	1.21	**0.97**

also report the results obtained using Bortfeldt's problems that have been recently proposed in [3]. He has defined 360 instances of strip-packing problems with 1000 rectangles and unknown optimal solutions. There are 12 sets of problems and 30 instances belonging to each set. They differ in four factors related to the objects to be placed: width, area, heterogeneity and maximum dimension ratio.

The hardware platform for the experiments was a PC Pentium IV, 2.66Ghz with 1024 MB RAM under Debian operating system. The algorithm has been implemented in C++.

4.2 Comparison with Low-Level Heuristics

The Table 1 shows the results, using Hopper's instances and allowing rotation, found by each single heuristic and the average and the best results obtained by our H-SP algorithm over 10 runs. In order to compare H-SP with low-level heuristics, we limited the running time of H-SP to 20 seconds.

The set H, in this test, is composed of the heuristics BLF,HR,BFDH*,BF, in their original versions [2]. And the set of parameters P is composed of all combinations of types of ordering (7) and types of rotation (4) for the remaining objects.

Each low-level heuristic is evaluated with each parameter in P ($7 \times 4 = 28$) and the best solution is shown, the time for each instance is not superior to 1 second. The results are calculated as the percentage from the optimal solution ($gap(\%) = \frac{solution-opt}{opt}$).

The quality of the solution found by the low-level heuristics has been strongly improved by the final constructive algorithm X_F given by our framework. The execution time of X_F is comparable to the execution time of low-level heuristics

[2] Originally each heuristic can decide when rotate or not an object, for the case of no rotation allowed instances, this functionality is not used.

Table 2. Average use of low-level heuristics in H-SP

Class	**Low-level heuristics**			
	BLF	HR	BFDH*	BF
C1	9.24	30.00	54.45	6.27
C2	11.14	27.17	11.95	49.72
C3	29.71	15.80	0.23	54.24
C4	34.46	24.41	8.52	32.59
C5	40.19	10.76	3.70	45.33
C6	15.82	10.97	1.95	71.25
C7	99.68	0.31	0	0
Average	34.48	13.07	5.49	46.94

Table 3. Gap to the solution for Hopper's instances with rotation allowed (RF)

Class	GA+ BLF	SA+ BLF	HRalg	SPGAL		**H-SP**100s		**H-SP**1000s	
				Average	Best	Average	Best	Average	Best
C1	4	4	8.33	1.7	1.7	**0**	**0**	**0**	**0**
C2	7	6	4.45	0.9	**0**	**0**	**0**	**0**	**0**
C3	5	5	6.67	2.2	2.2	2.22	2.22	1.78	**1.11**
C4	3	3	2.22	1.4	**0**	1.67	1.67	1.67	1.67
C5	4	3	1.85	**0**	**0**	1.11	1.11	1.11	1.11
C6	4	3	2.5	0.7	**0.3**	1	0.83	0.83	0.83
C7	5	4	1.8	0.5	**0.3**	1.03	0.97	0.69	0.56
Average	4.57	4	3.97	1.06	**0.64**	1	0.97	0.87	0.75

(in C7 instances, X_F and BLF take $0.0045s$ and $0.0035s$, respectively, to construct a solution).

In Table 2, we report the *average percentage of pieces* that each heuristic of the set H places in the final constructive algorithm X_F for each kind of problem.

We can remark that each problem requires a different combination of the low-level heuristics. This is the advantage of the implicit natural adaptation of the hyperheuristic framework. We remark that BFDH* tends to be less applied as the size of the problem increases, while BLF shows the exact contrary behavior. A pattern cannot be identified for both BF and HR heuristics. Note however that BF has been used more frequently than HR. In addition, HR is more useful in solving smaller problem categories. Thus, the application percentage of the low-level heuristics depends on the problem instance to be solved. Furthermore, the algorithm is able to self-adapt to the problem at hand.

Figure 6 shows a typical final constructive algorithm and its solution for a class C7 instance (especifically the C72 instance).

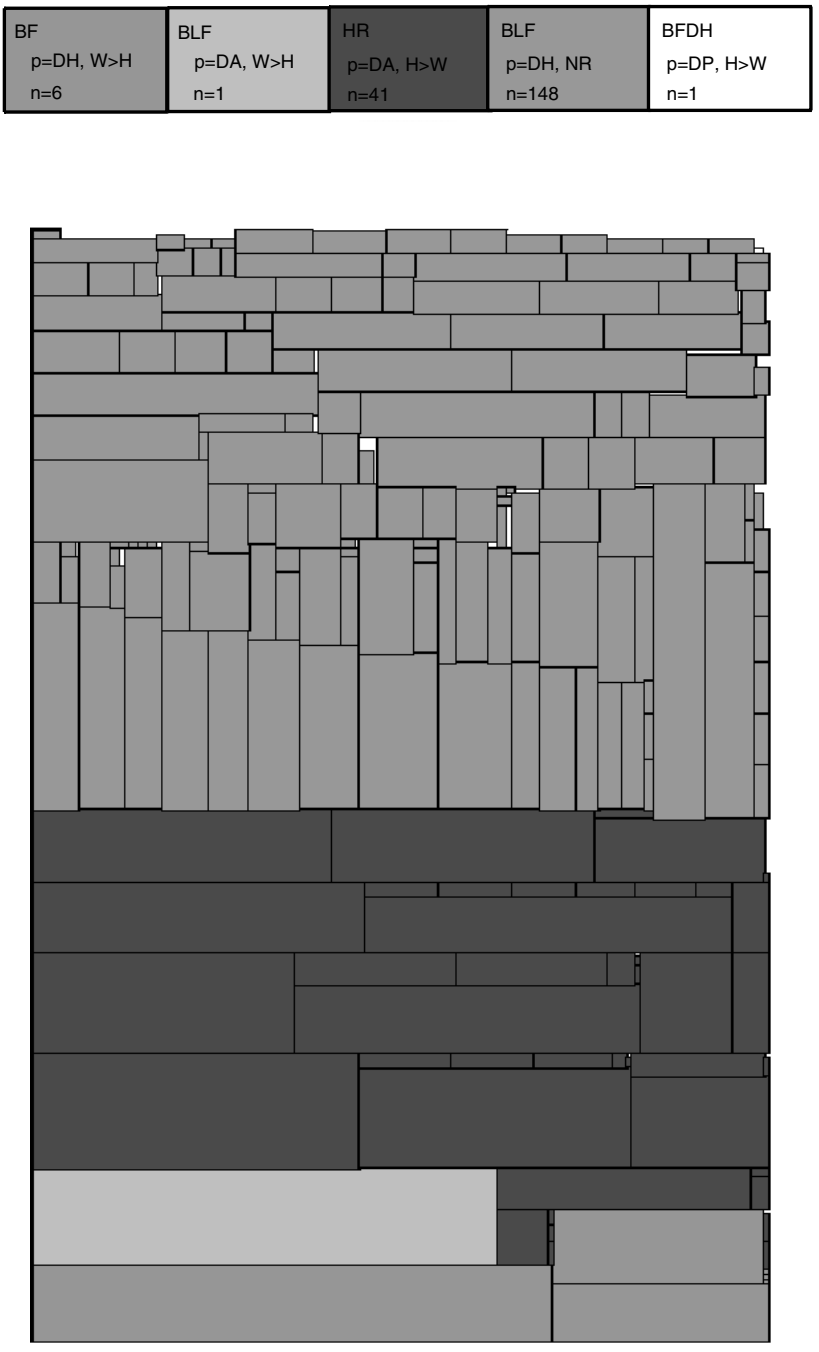


Fig. 6. Solution for instance C72

Table 4. Gap to the solution for Hopper's instances without rotation (OF)

Class	Iori algorithm	BF+ SA	SPGAL Best	GRASP Average	GRASP Best	**H-SP**100s Average	**H-SP**100s Best	**H-SP**1000s Average	**H-SP**1000s Best
C1	1.67	0	1.67	**0**	**0**	1.33	**0**	1	**0**
C2	2.22	6.25	2.22	**0**	**0**	**0**	**0**	**0**	**0**
C3	2.22	3.33	3.33	**1.11**	**1.11**	2.22	2.22	2.22	2.22
C4	4.75	1.67	2.78	**1.67**	**1.67**	2.11	**1.67**	**1.67**	**1.67**
C5	3.93	1.48	1.48	**1.11**	**1.11**	1.18	**1.11**	1.26	**1.11**
C6	4.00	1.39	1.67	1.58	1	1.39	1.11	1.22	**0.83**
C7	—	1.77	1.25	1.39	1.25	1.08	**0.97**	1	**0.97**
Average	3.13	2.27	2.06	0.98	**0.88**	1.33	1.01	1.2	0.97

4.3 Comparison with State-of-the-Art Algorithms

Tables 3 and 4 summarize the best results found in the literature [1, 3, 4, 12, 13, 16, 18, 21], and the results obtained by our hyperheuristic for the Hopper's instances. The results are calculated as the percentage from the optimal solution ($gap(\%) = \frac{solution-opt}{opt}$).

Tests with rotation allowed (RF)

We have first studied the problems where the rotation of the rectangles is allowed. Table 3 shows the results found in the literature for some algorithms compared with H-SP. The algorithms GA+BLF and SA+BLF [12], were run on a Pentium Pro 200 MHz with an average time per run of 674 minutes for SA+BLF and 136 minutes for GA+BLF. The deterministic algorithm HRalg [21], was run on a 2.4GHz CPU, with an average time per run of 5.59 seconds (0 seconds for C1 instances, 36 seconds for C7). SPGAL [4] reports an average time per run of 159 seconds on a 2GHz Pentium and the algorithm was run 10 times for each instance. The H-SP algorithm have been run 10 times with execution times of 100 and 1000 seconds for each instance. The set H is composed of the heuristics BLF, HR, BFDH* and BF, in their original versions.

Results in Table 3 show that H-SP gives good quality solutions and even better solutions than various other algorithms for the problem (metaheuristics and heuristics) except for the SPGAL algorithm. This algorithm is especially designed for these benchmarks and evaluates all possible rotations for each object to be positioned.

Tests without Rotation (OF)

We have also tested the algorithms considering the same benchmarks, but without allowing object rotations. To this test, the set H is composed of the heuristics BLF, HR, BFDH* and BF, in their no-rotation-allowed versions. The set P is reduced to only order parameters (rotation have no sense).

Table 5. Gap to the solution for Bortfeldt's instances

Set of Problems	**Type RG** GA4	H-SP 100s	**Type OG** GA4	H-SP 100s	**Type RF** H-SP 100s	H-SP 1000s
1	**2.44**	3.39	**4.43**	4.89	1.44	1.01
2	**1.86**	1.92	3.79	**3.70**	0.99	0.74
3	2.61	**1.54**	3.07	**2.32**	1.26	1.07
4	2.34	**1.04**	2.85	**1.64**	0.75	0.62
5	**1.27**	3.11	**2.08**	4.12	1.07	0.82
6	**1.04**	1.67	**1.68**	2.38	0.76	0.61
7	1.87	**1.59**	2.39	**2.13**	1.60	1.46
8	**1.18**	1.51	**1.62**	1.92	1.08	0.92
9	3.03	**2.12**	4.34	**3.45**	1.25	0.76
10	1.78	**1.27**	1.67	**1.52**	0.52	0.38
11	1.87	**1.46**	2.45	**1.97**	1.32	1.12
12	1.83	**1.58**	2.12	**2.03**	0.61	0.54
Average	1.93	**1.85**	2.71	**2.67**	1.05	0.84

Table 4 shows the results found by some algorithms compared with H-SP. The GRASP algorithm has been run 10 times on a 2GHz Pentium, the stopping criterion is of 60 seconds. BF+SA [7] has been run 10 times on a 2GHz Pentium with a limit of 60 seconds per run. Iori et al. [13] algorithm was run 300 seconds on a Pentium III at 800Mhz. SPGAL [4] reports an average time per run of 160 seconds on a 2GHz Pentium and the algorithm was run 10 times for each instance. The H-SP algorithm have been run 10 times with execution times of 100 and 1000 seconds for each instance.

Up to now, GRASP was the best approach. We obtained better average results than GRASP in the two biggest classes (C6 and C7).

4.4 Tests with Bortfeldt's Instances

We performed three series of tests with the 360 large new random instances proposed by Bortfeldt and Gehring [4], subdivided in 12 sets of 30 instances. For all these instances, the optimal solution is not known. We use as performance index the gap with the continuous lower bound *clb* [4] ($gap(\%) = \frac{(best_{found} - clb)}{clb}$).

In Table 5 we have compared the Bortfeldt's algorithm GA4 (based on SP-GAL) with H-SP. In the second and third columns we consider the problems type **RG**, that requires guillotine cuttings and allows objects to be rotated. For these set of problems the average execution time of algorithm GA4 is 895 seconds on a 2GHz Pentium. For these guillotinable instances, the set H is composed of low-level heuristics that respect that guillotine constraint. The heuristics are only two: HR and BFDH* (Section 2.1). For each problem instance the hyperheuristic is run once with a maximum execution time of 100 seconds.

In the fourth and fifth columns we consider the problems type **OG**, that requires guillotine cuttings and where the orientation of the objects is fixed. We used the same low-level heuristics as for RG instances. The average execution time for Bortfeldt's algorithm is 717 seconds on a 2GHz Pentium. For each problem instance the hyperheuristic is run once with a maximum execution time of 100 seconds and the average results are shown.

We also considered the problems type **RF** shown in the last column, where guillotine cutting is not required and the objects may be rotated. The set H is composed of the heuristics: BLF, HR, BFDH* and BF, in their original versions. For each problem instance the hyperheuristic is run once with maximum execution times of 100 and 1000 seconds.

We can remark that we are competitive for all these RG and OG benchmarks with Bortfeldt's algorithm. Moreover with the type RF we can see that we reduced the gap obtained for the RG and OG problems. This behavior was expected, since RF problems are less constrained, nevertheless, Bortfeldt and Gehring say that their algorithms (GA-4 is the best of them) obtain negligible improvements when they are applied with the post-optimization process [4], in other words, when they are applied to RF problems.

Our framework is flexible: we only had to change the set of low-level heuristics in each case, and the framework gives us competitive results.

5 Conclusions

This research allows us to conclude that using a hyperheuristic approach can improve the performance of single greedy heuristics. Moreover, the hyperheuristic is able to adapt itself to the problem by selecting a good combination of these low-level heuristics. This framework is quite general: we have shown that it could solve different strip packing problems (RF, OF, RG, OG). For solving a new problem type, the major task is the selection of suitable and efficient low-level heuristics. The hyperheuristic framework will allow cooperation among them, hopefully improving their single behaviors.

For future works, we believe that adding new operations and low-level heuristics can obtain configurations that explore in a better way the search space.

Acknowledgments

This work was carried out in the context of the Chile-France INRIA/CONICYT collaboration project.

References

1. Alvarez-Valdes, R., Parreño, F., Tamarit, J.M.: Reactive grasp for the strip packing problem. In: Proceedings Metaheuristic Conference MIC (2005)
2. Baker, B.S., Coffman, E.G., Rivest, R.L.: Orthogonal packings in two dimensions. SIAM Journal on Computing 9, 846–855 (1980)

3. Bortfeldt, A.: A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research 172, 814–837 (2006)
4. Bortfeldt, A., Gehring, H.: New large benchmarks for the two-dimensional strip packing problem with rectangular pieces. In: IEEE Proceedings of the 39^{th} Hawaii International Conference on Systems Sciences (2006)
5. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern search technology. Handbook of Meta-heuristics 16, 457–474 (2003)
6. Burke, E., Kendall, G., Whitwell, G.: A new placement heuristic for the orthogonal stock-cutting problem. Operations Research 52, 697–707 (2004)
7. Burke, E., Kendall, G., Whitwell, G.: Metaheuristic enhancements of the best-fit heuristic for the orthogonal stock cutting problem. Technical report, Univ. of Nottingham, Computer Science Technical Report No. NOTTCS-TR-2006-3 (2006)
8. Chazelle, B.: The bottom left bin packing heuristic: an efficient implementation. IEEE Transactions on Computers 32, 697–707 (1983)
9. Cowling, P., Kendall, G., Han, L.: An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: Congress on Evolutionary Computation, CEC 2002, pp. 1185–1190 (2002)
10. Cowling, P., Kendall, G., Soubeiga, E.: Hyperheuristics: A robust optimisation method applied to nurse scheduling. 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. 7–11. Springer, Heidelberg (2002)
11. Hopper, E.: Two-Dimensional Packing Utilising Evolutionary Algorithms and other Meta-Heuristic Methods. PhD. Thesis Cardiff University (2000)
12. Hopper, E., Turton, B.C.H.: An empirical investigation on metaheuristic and heuristic algorithms for a 2d packing problem. European Journal of Operational Research 128, 34–57 (2001)
13. Iori, M., Martello, S., Monaci, M.: Metaheuristic algorithms for the strip packing problem, pp. 159–179. Kluwer Academic Publishers, Dordrecht (2003)
14. Coffmann Jr., E., Garey, D., Tarjan, R.: Performance bounds for level oriented two-dimensional packing algorithms. SIAM Journal on Computing 9(1), 808–826 (1980)
15. Lesh, N., Marks, J., Mahon, A.Mc., Mitzenmacher, M.: Exhaustive approaches to 2d rectangular perfect packings. Information Processing Letters 90, 7–14 (2004)
16. Lesh, N., Marks, J., Mahon, A.M., Mitzenmacher, M.: New heuristic and interactive approaches to 2d rectangular strip packing. ACM Journal of Experimental Algorithmics 10, 1–18 (2005)
17. Lesh, N., Mitzenmacher, M.: Bubble search: A simple heuristic for improving priority-based greedy algorithms. Information Processing Letters 97, 161–169 (2006)
18. Martello, S., Monaci, M., Vigo, D.: An exact approach to the strip-packing problem. INFORMS Journal of Computing 15, 310–319 (2003)
19. Mumford-Valenzuela, C., Vick, J., Wang, P.Y.: Heuristics for large strip packing problems with guillotine patterns:An empirical study, pp. 501–522. Kluwer Academic Publishers, Dordrecht (2003)
20. Neveu, B., Trombettoni, G., Araya, I.: Incremental move for strip-packing. In: Proceedings of ICTAI 2007, Patras, Greece (2007)
21. Zhang, D., Kang, Y., Deng, A.: A new heuristic recursive algorithm for the strip rectangular packing problem. Computers and Operations Research 33, 2209–2217 (2006)

Probability-Driven Simulated Annealing for Optimizing Digital FIR Filters

Emmanuel Boutillon, Christian Roland, and Marc Sevaux

Université Européenne de Bretagne
UBS - Lab-STICC - Centre de Recherche
F-56321 Lorient – France
marc.sevaux@univ-ubs.fr

Summary. In this paper, we propose to mimic some well-known methods of control theory to automatically fix the parameters of a multi-objective Simulated Annealing (SA) method. Our objective is to allow a decision maker to efficiently use advanced operation research techniques without a deep knowledge of this domain. Classical SA controls the probability of acceptance using an a priori temperature scheduling (Temperature Driven SA, or TD-SA). In this paper, we simply propose to control the temperature using an a priori probability of acceptance scheduling (Probability Driven SA, or PD-SA). As an example, we present an application of signal processing and particularly the design of digital Finite Impulse Response (FIR) filters for very high speed applications. The optimization process of a FIR filter generally trades-off two metrics. The first metric is the quality of its spectral response (measured as a distance between the ideal filter and the real one). The second metric is the hardware cost of the filter. Thus, a Pareto-based approach obtained by a multi-objective simulated annealing is well suited for the decision maker. In this context, TD-SA and PD-SA method are compared. They show no significant differences in terms of performance. But, while TD-SA requires numerous attempts to set an efficient temperature scheduling, PD-SA leads directly to a good solution.

Keywords: Filter design, FIR, simulated annealing, multiobjective optimization, temperature regulation, feedback loop.

1 Introduction

Implementing a Simulated Annealing (SA) algorithm is quite an easy task and should be done in a few hours. But tuning the parameters for having good and interesting results is much more difficult. Most of the time, based on a set of instances (sometimes with known results), the parameters, one by one, are changed and set to their best values. Of course, interaction between the different parameters complicates the task.

What motivates this work is to let a decision maker (who often is not a specialist in optimization, and even less in tuning SA parameters) use the solver with a minimum number of comprehensive parameters. To achieve this goal, we try to translate the classical SA parameters to what could be easily understood

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 77–93, 2008.
springerlink.com

by the decision maker: a probability function and a number of iterations (a total running time).

In this paper, we consider that the temperature is controlled by a feedback loop. The feedback is given by the difference between the estimated probability of acceptance at a given iteration number and the desire probability of acceptance at this moment. This technique is applied on a signal processing problem: the joint optimization (i.e. multiobjective function) of the performance of a numerical Finite Impulse Response (FIR) filter and its related hardware complexity. Related work is presented in [4]. Note that the FIR filter is one of the key tools of the signal processing domain. The domain of application of FIR filter is thus very large (radar, sonar, communication, sysmography, ...).

The rest of the paper is divided in five sections. Section 2 describes the problem of FIR filter design and the relative metrics associated to the FIR filter performances and its hardware cost. Section 3 proposes a literature review of known works in the same area, followed by the proposed approach in section 4. Numerical experiments are conducted in section 5 before a conclusion in the last section.

2 The Digital FIR Filter Problem Design

This section presents the problem of digital FIR filter design for a high speed dedicated architecture. After recalling the definition of a FIR filter, the classical design flow is given. Then, an alternative method is proposed and the cost function of performances and complexity are presented. General information can be found in [8,12,13].

2.1 Definition of a FIR Filter

A FIR filter is a common tool in signal processing. The input signal of a FIR filter is a numerical series (typically, the samples of a captor) $x(n)$ indexed by an integer n. Generally, the signal of interest is corrupted by noise or other non significant signals. The FIR filter processes the input signal $x(n)$ and generates a filtered output signal $y(n)$ that rejects part of the jamming signal and noise. A FIR filter of order N is characterized by its finite impulse response (FIR) of length N given by $H = (h(0), h(1), ..., h(N-1)$ [1]. The output $y(n)$ at time n of the filter H is given by the equation:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{1}$$

This operation is noted $y(n) = h(n) * x(n)$, where $*$ stands for convolution.

The coefficients H of the filter are invariant over time and identical to the impulse response of the filter (see Figure 1-a).

In signal processing theory, filters are characterized by their frequency response. The frequency response $H(f)$ is obtained with the Fourier Transform

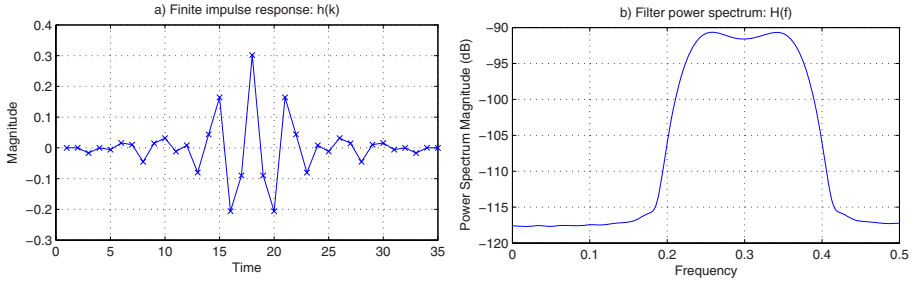


Fig. 1. Representation of a FIR filter

(FT) of the finite impulse response (see Figure 1-b). In the sequels, the phase response will not be considered and we only get focused on the amplitude response (i.e. $|H(f)|$) of the FIR filter (the majority of FIR application in signal processing).

2.2 The Problem of FIR Filter Synthesis

The classical process of FIR synthesis is divided in 3 stpdf: first, the ideal filter is defined according to the spectral characteristic of the signal and the target of the application. In general, this filter has cliff transition and this results in an infinite impulse response. In order to obtain an implementable filter, the filter constraints are relaxed and the template of an acceptable filter is defined. For example, a template of band-pass filter is defined by several parameters: the bounds of the passband frequencies (fi_1, fi_2), the absolute value of the maximum gain in the passband frequencies, the size of the transition bands (f_1, fi_1 and f_2, fi_2) and the rejection factor in the rejection band. Va_2 (resp. Va_3) is the maximum (resp. minimum) level for the passband. Va_1 is the maximum level for the rejection band (see Figure 2).

Given a template, the generation of H can be obtained by several methods: the Hamming method, the Hanning method, the Remez method, the Kaiser method and the window method to cite some of the most popular[1] [8].

All those methods provide real values of H. The next step is then to represent the real value in a fix precision format for the implementation. This task can be tricky because quantization impacts on both performance and hardware complexity. Some papers work research an optimal design of FIR with only one constraint. The effort is concentrated on the quantization of the filter coefficients to obtain H_q, the quantized impulse response (the value of H_q are then integer). In the general case, the authors have one objective: how to limit the degradation of the frequency performance between the real and the quantized FIR filter.

[1] These methods are all available in Matlab.

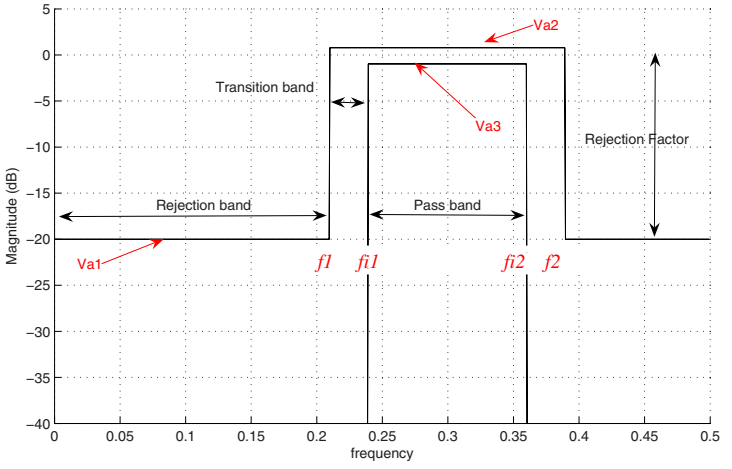


Fig. 2. Template of a FIR filter

2.3 Proposed FIR Designs

In this paper, we propose to proceed in a different way than the straightforward method defined above. The idea is to start from a classical method to generate H, then quantize H to obtain H_q^{init} and then, directly optimize H_q^{init} by considering jointly the performance and the complexity of the filter. This approach is quite new and few recent papers proposed to perform FIR design in a similar way [2,7,10,11,14]. Note that except for [14], all those papers are single objective optimization methods, and the bi-objective proposal concerns different criteria using genetic algorithms.

Those techniques require two types of metrics: a measure of the performance of the filter and a measure of the hardware complexity of the filter. Since the details of FIR design problem is out of the scope of the paper, the next two sections describe briefly the algorithm that will be used to compute those two metrics.

2.4 Performance Measures

The cost function C between the template $G(f)$ and the actual filter $H_q(f)$ defines the "quality" of the design. Ideally, when $C(H_q, G) = 0$, then $H_q(f)$ is inside the template for all frequencies of the application. In the following, we will consider that, for a given frequency f, if $H(f)$ is inside the template, then $C(H_q(f), G(f)) = 0$, otherwise, $C(H_q(f), G(f))$ is equal to a weighted distance of the actual response $H_q(f)$ and the closest limit of the template for the frequency f.

In practice, the cost function of the template is given by the summation of $C(H_q(f_k), G(f_k))$, $f_k = k/N_{fft}$ for $f = 0 \cdots N_{fft} - 1$. The N_{fft} value of $H_q(f_k)$

Algorithm 1. Template cost function

initialization step:
  fix parameters cost function (N, N_{fft}, Va_1, Va_2 and Va_3)
  fix band-limited and stop-band frequencies (f_1, f_2, fi_1 and fi_2) `// (see Figure 2 for some of these parameters)`
find a new solution h_i with SA algorithm
calculate: $H = log|fft(h_i)|$
$d_1 = d_2 = d_3 = d_4 = 0$
i = 1
repeat
  if $H(i) > Va_1$ **then**
    $d_1 = d_1 + (H(i) - Va_1)^{\alpha}$
  endif
  i = i+1
until $i < f_1$
repeat
  if $H(i) > Va_2$ **then**
    $d_2 = d_2 + (H(i) - Va_2)^{\alpha}$
  endif
  if $i > fi_1$ *and* $i < fi_2$ *and* $H(i) < Va_3$ **then**
    $d_3 = d_3 + (Va_3 - H(i))^{\alpha}$
  endif
  i = i+1
until $i < f_2$
repeat
  if $H(i) > Va_1$ **then**
    $d_4 = d_4 + (H(i) - Va_1)^{\alpha}$
  endif
  i = i+1
until $i < fft\_size/2$
$d = d_1 + d_2 + d_3 + d_4$

are obtained by a Fast Fourier Transform of H_q on N_{fft} points. The details on the computation of the distance is provided in Algorithm 1.

2.5 Hardware Complexity of a Filter $H_q(z)$

In this paper, we assume that the hardware is dedicated to the filter and that the input rate of the filter is equal to the clock frequency of the hardware. In other words, an output sample is computed every clock cycle. The architecture requires N hardware multipliers, each one dedicated to a specific multiplication with a fix constant (i.e. $h_q(k)$ for the k^{th} multiplier). The architecture of the FIR filter is presented Figure 3.

Roughly speaking, the complexity of the architecture is composed of three terms, the first cost is related to the register required to store the N previous received input samples $(x(n-k))_{k=0,\ldots,N-1}$, the second term is the total cost of the N multipliers and the last term is the cost of the final adder to sum

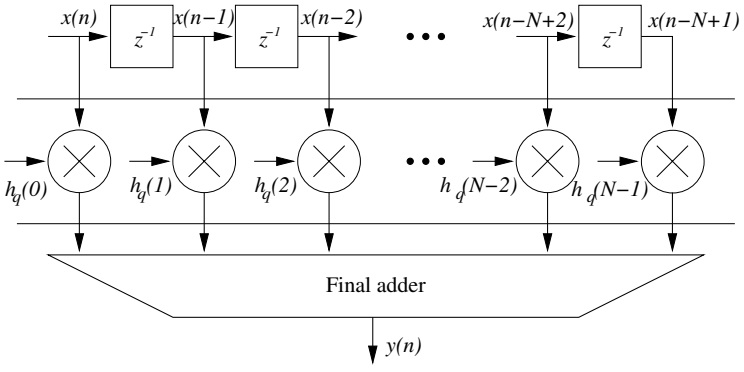


Fig. 3. Architecture of a FIR filter

the N partial results $(h_q(k)x_{n-k})_{k=0,\ldots,N-1}$. The first and third terms can be assumed independent of the implementation (it is a constant cost). The second term depends on the implementation choice and on the value of the coefficient $h_q(k)_{k=0,\ldots,N-1}$ (variable term). This third term alone will be used as a metric for the hardware complexity of the design. The hardware cost depends on the binary representation of the coefficients. The exact formulation of the hardware cost is out of the scope of then paper but is non-linear. To give a brief idea, a multiplication by 31, 32, 33 has a complexity of 5, 1, and 2 respectively, *i.e.* the number of non-zero elements in the binary representation. Using the Canonical Signed Digit representation (a more efficient representation of number, where, for example, 31 is coded as $32-1$ [7]), the hardware cost function can be computed by Algorithm 2.

In the following sections, we consider the cost function C^{csd}, defined as $C^{csd} = \sum_{k=0}^{N-1} C^{csd}_{h_q(k)}$, where $C^{csd}_{h_q(k)}$ is the number of non zero value of the coefficient $h_q(k)$ in the CSD representation. This number is given by Algorithm 2 where C is the CSD decomposition of x and C^{csd}_x the number of non-zero values of C.

3 Description of the Proposed Approach

The method itself is rather simple. Everything is based on the multiobjective simulated annealing scheme where the temperature is controlled by a feedback loop.

3.1 Multiobjective Optimization by Temperature Driven Simulated Annealing (TD-SA)

We briefly recall in this section the multiobjective optimization framework using a simulated annealing algorithm.

When m objectives $f_i,\ i \in [1, m]$ are simultaneously considered for minimization, we need to define the concept of Pareto dominance. Instead of giving an

Algorithm 2. Complexity (CSD decomposition of h_q)

```
1  initialization step:
2    C^csd = 0
3    Nb = 12 // Number of bits for quantization
4  for i = 1 to length(h_q) do
5  |   c = 0
6  |   x = |h_q(i)|
7  |   for k = Nb-1 to 1 (step -1) do
8  |   |   M = 2^(k-1)
9  |   |   if x > M/2 then
10 |   |   |   x = M - x
11 |   |   |   c = c + 1
12 |   |   endif
13 |   endfor
14 |   C^csd = C^csd + c
15 endfor
16 return C^csd
```

absolute value for a solution, a partial order is defined based on dominance. A solution is said to dominate another solution when it is better on one objective, and not worse on the other objectives. Thus a solution x dominates a solution y if and only if $\exists i \in [1,m] : f_i(x) < f_i(y)$ and $\forall j \in [1,m], j \neq i : f_j(x) \leq f_j(y)$. A solution is said to be non-dominated if no solution can be found that dominates it.

The definition of the dominance relation gives rise to the definition of the Pareto optimal set, also called the set of non-dominated solutions. This set contains all solutions that balance the objectives in a unique and optimal way. The aim of multi-objective optimization is to induce this entire set. Picking a single solution from this set is then an a posteriori judgement, which can be done in terms of concrete solutions with concrete trade-offs, rather than in terms of possible weightings of objectives. The question for multiobjective optimization is now how to find this Pareto optimal set.

We recall here that our objective is not to design the best possible multiobjective algorithm for solving the filter design problem but to propose an easy implementable solution.

Based on previous assumptions, we use the work of Nam and Park [9] and the multiobjective algorithm will be based on simulated annealing. The literature on this topic is important. A good introduction on evolutionary algorithms for multiobjective optimization can be found in [3,5].

Algorithm 3 presents a basic multiobjective simulated annealing framework. In the *initialization step*, as in all neighborhood search metaheuristics, an initial solution should be provided. Fixing the annealing schedule (Alg 3, line 3) and setting the initial temperature (Alg 3, line 4) are two empiric tasks that partially motivate the work presented in this paper. For general explanations on standard settings, we refer the reader to the original paper [9].

Algorithm 3. Basic multiobjective simulated annealing

```
1  initialization step:
2    find an initial solution x
3    fix an annealing schedule 𝒯
4    set initial temperature T = T_0
5  repeat
6  |   neighborhood search:
7  |     find a solution x' ∈ 𝒩(x)
8  |   if x does not dominates x' then
9  |   |   x' ← x
10 |   else
11 |   |   determine ΔC (the variation of the cost function)
12 |   |   draw p ~ 𝒰(0,1)
13 |   |   if e^{-ΔC/T} > p then
14 |   |   |   x' ← x
15 |   |   endif
16 |   endif
17 |   update temperature T according to 𝒯
18 until stopping criterion satisfied
```

The general loop (Alg 3, lines 3-3), as in all metaheuristic algorithms, does not differ much from a single objective simulated annealing. First a neighbor solution x' of current solution x is randomly generated. In single objective optimization, if the new solution x' is better than x, it is accepted as the new current solution. In the multiobjective case, it is accepted if x' is not dominated by x (Alg 3, line 8), which means if it is not worse than the current solution.

Now, when x' is dominated by x, as in the classical simulated annealing algorithm, x' can become the new current solution (in order to escape local optima) under a probability condition. In [9], the authors call it *probability transition* and expose six different criteria. We use the random cost criterion. ΔC (Alg 3, line 11) is computed by Equation 2.

$$\Delta C = \sum_{j=1}^{m} \beta_j (f_j(x') - f_j(x)) \tag{2}$$

where β_j is a random variable with uniform probability distribution. Acceptance probability is given by the so-called Boltzmann's equation (see [1] for more information).

At the end of the loop iteration, the temperature is updated according to the cooling schedule. Classical cooling schedules refer to geometric evolution of the temperature $T_k = \alpha^k T_0$, where T_k is the temperature at iteration k, α is the cooling rate ($0 < \alpha < 1$) and T_0 the initial temperature.

3.2 Parameter Reduction in SA – from TD-SA to PD-SA

In the description of the TD-SA (Algorithm 3), the following parameters have to be set by the decision maker or by the end-user of the solver:

- the initial solution x
- the annealing schedule $\mathcal{T}$
- the initial temperature T_0
- the stopping conditions

The initial solution x can be generated randomly or provided by the decision maker (from previous runs of the solver, or from experience). If the annealing schedule follows the description of the geometric evolution of the previous section, parameter α should be provided, as well as the initial temperature T_0 and the stopping conditions of the algorithm.

The stopping conditions are easy to set. Usually, a running time is wished by the decision maker and is converted into a maximum number of iterations. But the initial temperature T_0 and the cooling rate α requires several attempts for a good setting. Moreover, this setting should be done again when the set of instances changes.

To avoid this, we propose to replace the annealing schedule by a function representing at each iteration the probability of accepting a non-improving move. This new method will be called Probability-Driven Simulated Annealing (PD-SA).

Figure 4 draws such a simple potential function. In a solver with a graphical user interface, the function could be chosen from a library. In our case, we use $p(x) = P_0 \times (1 - 5/\texttt{ItMax})^x$ as a sample function. P_0 is the probability of accepting a non-improving solution at the beginning of the search. P_0 and `ItMax` are incorporated into the library (in Figure 4, $P_0 = 0.3$, and $\texttt{ItMax} = 100$). Note that the value of $P_0 = 0.3$ can be fixed for once, thus, the only parameter of the system simply becomes `ItMax`.

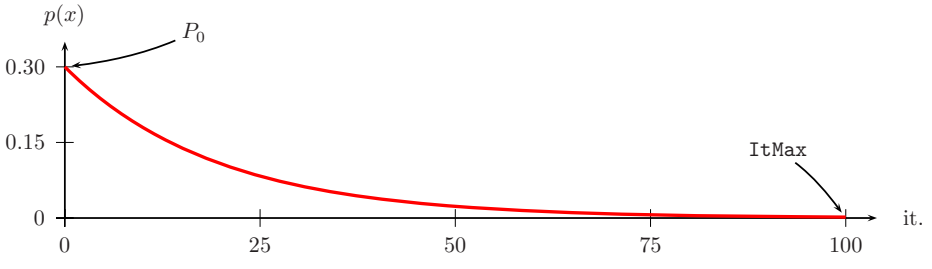


Fig. 4. A potential probability function

A decision maker is able to fully understand the purpose of this function and its two parameters. In order not to change the simulated annealing algorithm, we will transform the current probability (the one at the current iteration) into a temperature for the SA algorithm (see next section). For the classical SA algorithm, T_0 can be reversely computed from P_0 with the first non-improving move of the algorithm ($P_0 = e^{-\Delta C/T_0} \Leftrightarrow T_0 = -\Delta C/ln(P_0)$).

3.3 Controlling the Temperature Parameter in PD-SA

With the probability scheme proposed in the previous section, the major question coming is "why keeping a temperature-based system if we know in advance the probability for accepting non-improving moves?". First, removing the temperature will remove a degree of freedom in the SA approach, second, removing the temperature will change the SA scheme and will need partial re-writing of the program. Instead, we will use a feedback loop as in automatic control to keep a temperature as close as possible of the desired temperature (*i.e.*, of the probability value provided by the probability function).

In a classical TD-SA scheme, each time a new generated solution has an objective function value worse than the current solution (i.e. a cost function $C > 0$), this degrading solution is accepted if the value of $e^{-\Delta C/T}$ is greater than an uniform random value between 0 and 1. The event of accepting a degrading solution as exactly a probability of $\tilde{p} = e^{-\Delta C/T}$ to occur.

In the new PD-SA approach, we keep the same process to accept or not a degrading solution. The main difference is that, instead of cooling blindly the temperature in a deterministic way ($T(i+1) = \alpha \cdot T(i)$), we try to control the temperature T_c so that $\tilde{p}(i) = e^{-\Delta C/T_c}$ equals exactly $p(i)$ the desire probability of acceptance at iteration number i. Thus, each time a non-improving solution is generated, 3 cases can occur:

1. $\tilde{p}(i) > p(i)$, than the probability of acceptance is too high and T_c should be decreased.
2. $\tilde{p}(i) = p(i)$, than the probability of acceptance is good and T_c is correct.
3. $\tilde{p}(i) < p(i)$, than the probability of acceptance is too low and T_c should be increased.

To perform such an update, the temperature is adjusted using a feedback loop as:

$$T_c = T_c + T_c\epsilon(p(i) - \tilde{p}(i)) \tag{3}$$

where ϵ is a parameter of the feedback loop that weights the correction factor (ϵ is set to 1 in our simulation).

One can note that this feedback loop is similar to the Proportional Integral (PI) corrector [6] if we consider T_c in the log domain. In fact:

$$log(T_c) = log(T_c) + log(1 + \epsilon(p(i) - \tilde{p}(i)) \tag{4}$$

and thus:

$$log(T_c) \approx log(T_c) + \epsilon(p(i) - \tilde{p}(i)) \tag{5}$$

The PI corrector is well known in automatic control system to be a very robust corrector, with no bias and generally stable for a large range of values of ϵ. Performing this kind of feedback loop in unusual in metaheuristics but its efficiency has been proved in automatic control systems since a very long time.

One can note an advantage of this type of feedback control compared to the classical cooling temperature scheme. In fact, if the local solution is a local

minima and if the current temperature is too low, then the SA algorithm is trapped in this local minima until the end of the SA process. On the contrary, with the feedback loop, in such a situation, the temperature will increase in a geometrical way so that, at one moment, it will escape this local minima and restart a worthwhile research process.

Numerical experiments will show the difference between PD-SA and TD-SA.

4 Numerical Experiments

This section presents the comparison between TD-SA and PD-SA. The parameters of the SA algorithms are first presented (coding, neighborhood and initial solution). Then computational results are presented and discussed.

4.1 Coding, Neighborhood and Initial Solution

A solution is represented by 33 integer values (stored in an array, $s[i], i = 1, \ldots, 33$). Each value belongs to the range $[-1023, +1023]$, corresponding to the physical size of the component (a power of 2). These values are the coefficients of the filter *i.e.*, the impulse response of the filter. To obtain the frequency response of the filter, it is necessary to use a FFT. The impulse response is symmetric so the values in the array are also symmetric ($s[i] = s[33 - i + 1]$) and the sum of coefficients should be constant (the square root of the energy remains the same along the search).

The neighborhood is defined so that we respect the constraints on the encoding. To move from one solution to a neighbor, one of the coefficients is either increased by one or two or decreased. The symmetric value is changed accordingly. To keep the constant sum property, another symmetric pair of coefficients is inversely modified.

An initial solution for the proposed method is obtained from classical design filters, *e.g.* Hanning window multiply iFFT of ideal filter. This solution is usually blindly used by practitioners for designing filters. By doing so, we ensure the designer to have at least a feasible and "classical" solution.

4.2 Computational Results

Numerical experiments have been conducted in order to assert that the proposed method is at least as good as a classical simulated annealing algorithm. To do so, several figures are presented and commented below.

Even if the general purpose of the algorithm is to reduce the number of parameters, some of them are necessary. First, the number of coefficients of the filter –sometimes considered as a meta-parameter– is left to the decision maker. For the experiments, we set it to the most commonly used value, *i.e.* 33 coefficients (*e.g.* like for a standard FIR1 function). Second, the size of the FFT is usually set to 2048 values. Then, several parameters presented in the cost function

(see parameters Va_1, Va_2, Va_3 in Algorithm 1) and in the design of the template (see parameters f_1, f_2, fi_1, fi_2 in Figure 2) have to be chosen by the decision maker. In figure 2, some of them depend on the final application. We propose in the future to develop a graphical user interface that will help the designer to set these parameters. When it is not explicitly mentioned, the maximum number of iterations is 50 000 corresponding to a reasonable amount of time (5 minutes).

For all the experiments, we try to compare the execution of the standard simulated annealing algorithm and the version with automatic temperature setting. On one hand, simulation operates with an initial temperature T_0, and on the other hand, the parameter used at the beginning of the search is P_0, the initial probability of accepting non-improving moves. As an example, we auto-magically set $T_0 = 30$ and arbitrarily set $P_0 = 0.3$.

First we compare the influence of the maximum number of iterations. Figure 5 compares the Pareto solutions of the two approaches obtained after different maximum number of iterations. These Pareto solutions are the best solutions over 250 different runs for each approach.

In these figures, Pareto solutions are presented for the two solution techniques, TD-SA and PD-SA when $T_0 = 30$. The initial solution blindly used by decision makers is also noted as "Initial".

Figure 5 shows just 3 particular cases of the Pareto sets obtained with the TD-SA and the proposed PD-SA. It did not give any insight of which technique is more efficient. In order to obtain an objective comparison between the two methods, we perform a series of "match racing" between TD-SA and PD-SA. Each match racing starts from the same initial condition and process the same number of iteration. Once the two Pareto function are obtained, they are merge to create a new Pareto function. Let N_{PD-SA} (respectively N_{TD-SA}) be the number of point of the PD-SA Pareto curve (respectively TD-SA Pareto curve) that are not dominated by a point of the TD-SA Pareto curve (respectively TD-SA Pareto curve). Then the result of the match racing is given by:

$$Q = \frac{N_{PD-SA}}{N_{PD-SA} + N_{TD-SA}} \tag{6}$$

Thus Q is a number between 0 and 1. $Q = 0$ means that TD-SA dominates PD-SA, $Q = 1$ means that PD-SA dominates TD-SA. Note also that is the Pareto curve of PD-SA and TD-SA are identical, then $N_{PD-SA} = N_{TD-SA}$ and thus $Q = 0.5$. We perform each time 250 match racing to obtain an estimation of the expectation of $E[Q]$ and the standard deviation σ_Q of Q. The values of $E[Q]$ are obtain with an marginal error of $\pm\, \sigma_E[Q] = \frac{\sigma_Q}{\sqrt{250}} = 0.025$ for $T_0 = 30$ and 0.01 for $T_0 = 1$. Table 1 gives the results of the match racing for 5000, 10000 and 25000 iterations. Tests have been conducted for two different values of the initial temperature T_0 (30 or 1) in TD-SA. It appears clearly that tunning correctly the temperature can lead to better results in TD-SA than in PD-SA. As already mentioned, an end-user has to manually tune the temperature and this might be long and difficult.

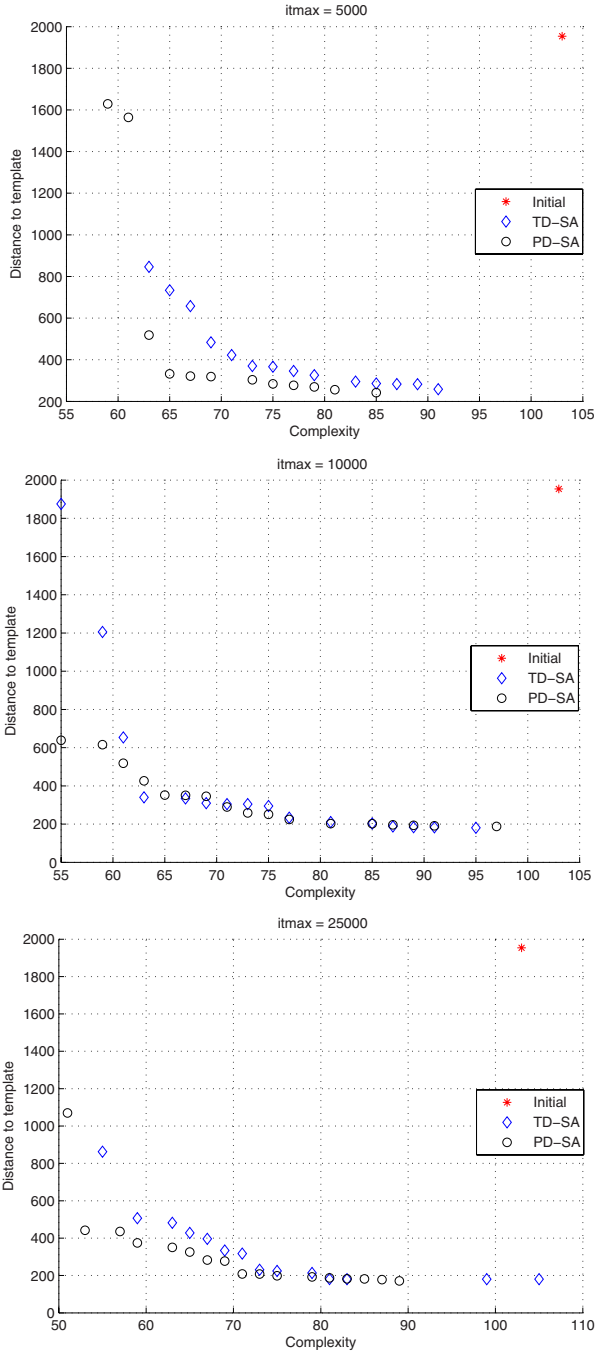


Fig. 5. Solutions after 5 000 it. (up), 10 000 it. (middle) and 25 000 it. (bottom).

Table 1. Dominance ratio (Q) of PD-SA over TD-SA

Iterations	5 000	10 000	25 000
$\boldsymbol{T_0 = 30}$			
Average	0.60	0.65	0.63
St.Dev.	0.40	0.37	0.36
$\boldsymbol{T_0 = 1}$			
Average	0.33	0.34	0.33
St.Dev.	0.13	0.13	0.12

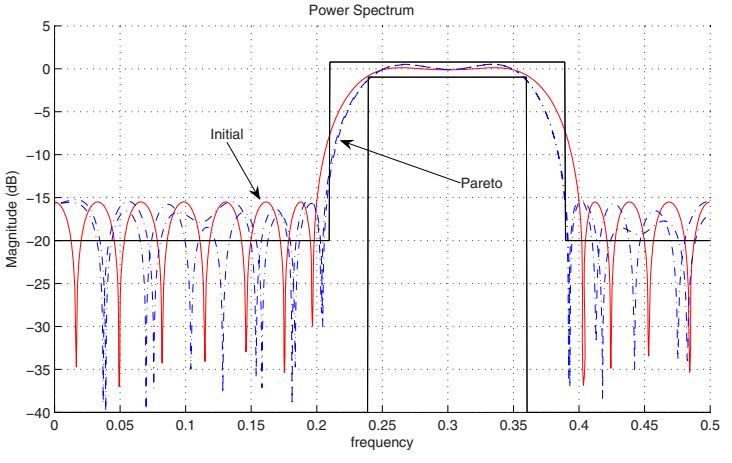


Fig. 6. Power spectrum

To be sure that the produced Pareto solutions are of good quality in terms of distance to template, we plot the initial solution and some Pareto solutions represented as power spectrum. Hence it is possible to draw the template on the same figure. In Figure 6 dashed lines represent these solutions whereas the solid line corresponds to the initial solution. It is clear in the figure that the Pareto solutions are largely better than the initial solution, especially the part of the response of the filter outside the transition band is much smaller for all Pareto solutions represented here.

We now compare the resolution of the TD-SA approach and the new PD-SA method (see Figure 7) after 50 000 iterations. Note that the constraints for the template have been tightened. For this figure, it is clear that none of the approaches is better than the other. But from the decision maker point of view, the PD-SA gives the *same* results without the parameter setting phase (to find the best values of the parameters needed in the TD-SA, more than 10 different attempts were necessary).

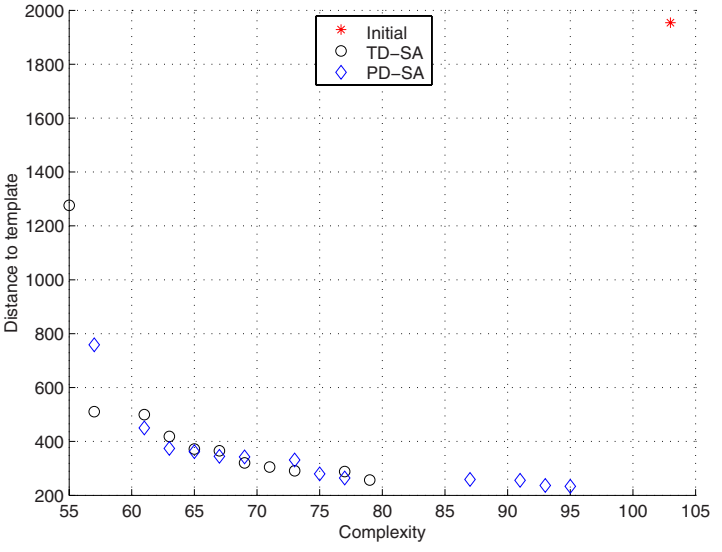


Fig. 7. Comparison of solutions

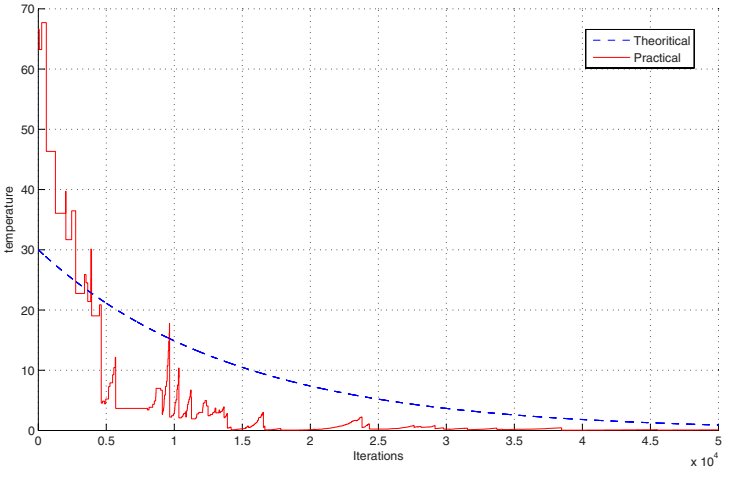


Fig. 8. Comparison of the temperature evolution in PD-SA

To show how the feedback loop influences the temperature, we draw in Figure 8 the theoretical temperature and the temperature computed afterwards in PD-SA. Since the practical temperature is computed not at every iteration, it results a stepwise curve that oscillate around the theoretical temperature.

5 Conclusion

In this paper, we have presented an alternative approach for the design of a digital FIR filter minimizing two objectives, namely the distance to a template and the complexity of the filter. We have to mentioned here that, to our best knowledge, it is the first time that these two objectives are dealt simultaneously in a approximate Pareto-based approach using simulated annealing.

In the field of the filter design, it is of course much better to present several solutions to the decision maker which finally can choose the most appropriate alternative for his application.

Numerical experiments do not show any advantage in terms of performances to the probability-driven simulated annealing method but no clear drawbacks either. Of course, the proposed approach contains less parameters that have to be set and represents a progress for non-specialist people in the field of optimization and for end-users.

Through this paper, the reader can notice that several other parameters need to be set, even in the new method, and are not always explicitly mentioned here. The reason is that a graphical user interface might help the decision maker to set these parameters by choosing general templates from libraries with best known values and/or experimented designer knowledge. Several templates will be also included in the library. For example, the number of coefficients of the filter (here set to 33) can become a "meta" parameter. In that case, a dedicated neighborhood procedure will be designed for finding the most adapted number of coefficients.

We believe that the proposed probability-driven simulated annealing approach can be extended with success in many other application domains and that this approach will help the spreading of advanced SA techniques in the engineering community.

References

1. Aarts, E.H.L., Korst, J.: Simulated Annealing and Boltzmann Machines. John Wiley, Chichester (1989)
2. Cen, L., Lian, Y.: Complexity reduction of high-speed fir filters using micro-genetic algorithm. In: First International Symposium on Control, Communications and Signal Processing, pp. 419–422 (2004)
3. Coello, C.: EMOO web pages, http://www.lania.mx/~ccoello/EMOO/
4. Damera-Venkata, N., Evans, B.L.: An automated framework for multicriteria optimization of analog filter designs. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 46(8), 981–990 (1999)
5. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & sons, New York (2001)
6. Johnson, M.A., Moradi, M.H. (eds.): PID Control. Springer, London (2005)
7. Kilambi, S.M., Nowrouzian, B.: A genetic algorithm employing correlative roulette selection for optimization of FRM digital filters over CSD multiplier coefficient space. In: IEEE Asia Pacific Conference on Circuits and Systems, 2006. APCCAS 2006, December 2006, pp. 732–735 (2006)

8. Mou, Z., Duhamel, P.: Fast FIR filtering: Algorithms and implementations. Signal Processing 13(4), 377–384 (1987)
9. Nam, D., Park, C.H.: Multiobjective simulated annealing: A comparative study to evolutionary algorithms. International Journal of Fuzzy Systems 2(2), 87–97 (2000)
10. Oner, M.: A genetic algorithm for optimisation of linear phase fir filter coefficients. In: Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers, November 1998, vol. 2, pp. 1397–1400 (1998)
11. Qiao, J., Fu, P., Meng, S.: A combined optimization method of finite wordlength fir filters. In: First International Conference on Innovative Computing, Information and Control, 2006. ICICIC 2006, August 2006, vol. 3, pp. 103–106 (2006)
12. Siohan, P., Benslimane, A.: Synthèse des filtres numériques non récursifs à phase linéaire et coefficients de longueur finie. Annales des Télécommunications 39(7-8), 307–322 (1984)
13. Siohan, P., Benslimane, A.: Finite precision design of optimal linear phase 2-D FIR digital filters. IEEE Transactions on Circuits and Systems 36(1), 11–22 (1989)
14. Thomson, R., Arslan, T.: An evolutionary algorithm for the multi-objective optimisation of VLSI primitive operator filters. In: Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC 2002, vol. 1, pp. 37–42 (2002)

RASH: A Self-adaptive Random Search Method

Mauro Brunato and Roberto Battiti

Dipartimento di Ingegneria e Scienza dell'Informazione
Università di Trento, via Sommarive 14, I-38100 Trento — Italy
{battiti,brunato}@dit.unitn.it

Summary. A variation of an adaptive random search algorithm for the optimization of functions of continuous variables is presented. The scheme does not require any assumptions about the function to be optimized, apart from the availability of evaluations at selected test points. The main design criterion of the Reactive Affine Shaker (RASH) scheme consists of the adaptation of a search region by an affine transformation. The modification takes into account the local knowledge derived from trial points generated with a uniform probability in the search region. The aim is to scout for local minima in the attraction basin where the initial point falls, by adapting the step size and direction to maintain heuristically the largest possible movement per function evaluation. The design is complemented by the analysis of some strategic choices, like the double-shot strategy and the initialization, and by experimental results showing that, in spite of its simplicity, RASH is a promising building block to consider for the development of more complex optimization algorithms.

Keywords: Stochastic search, adaptive random search, mathematical programming.

1 Introduction

Finding the global minimum of a function of continuous variables $f(\boldsymbol{x})$ is a well known problem for which substantial effort has been dedicated in the last decades, see for example the bibliography in [12]. Apparently, no general-purpose *panacea* method exists which can guarantee its solution at a desired accuracy within finite and predictable computing times. In fact, the different versions of the so called "no free lunch theorems" imply that "for any algorithm any elevated performance over one class of problems is paid for in performance over another class" [16].

On the other hand, most real-world optimization tasks are characterized by a rich correlation structure between candidate solutions which are close, in a suitable metric defined over the independent variables. Local search techniques capitalize on this local structure by postulating that a better solution can usually be found in the *neighborhood* of the current tentative solution. In this manner, after starting from an initial configuration of the independent variables $\boldsymbol{x}^{(0)}$, a *search trajectory* of a discrete dynamical system is generated, in which point $\boldsymbol{x}^{(t+1)}$ is chosen in the neighborhood of point $\boldsymbol{x}^{(t)}$. Under suitable conditions (e.g., lower-bounded function, decreasing values of $f(\boldsymbol{x}^{(t)})$ during the search

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 95–117, 2008.
springerlink.com

with a sufficiently fast rate of decrease) the trajectory will converge at a *local minimizer*. The set of initial points which are mapped to a specific local minimizer by the local search dynamical system is called the *basin of attraction* of the minimizer.

Many recent global optimization techniques deal with ways to use a local search technique without being trapped by local minimizers, notably the Simulated Annealing technique based on Markov chains, see for example [5] and [13].

Because of the growing awareness that no single general-purpose method can be efficiently applied to different problems, recent research considers the appropriate integration of basic algorithmic building blocks, like various stochastic local search techniques [11], the so-called meta-heuristic techniques [7], the various combinations of genetic operators [8], the *algorithm portfolio* proposals [9]. The crucial issue is that of tailoring the appropriate combination of components and values of critical parameters, a process that implies an expensive learning phase by the user and that can be partially automated by machine learning techniques, as it is advocated in the reactive search framework [2], see also the web site `www.reactive-search.org`.

Research and applications demand a careful design of each component, which should be studied in isolation before considering integration in more complex schemes. In this manner, the added value of the combination w.r.t. the components can be judged in a statistically sound manner.

In this paper we focus on a "direct method" for optimization [10] which considers only function evaluations. We furthermore assume *no a priori knowledge* about f, the knowledge will be only that acquired during evaluations $f(\boldsymbol{x})$ at different values of the independent parameters. In particular we develop an algorithm based on the stochastic (or random) local search framework originally proposed in [14]. We state the main design criteria and study some critical choices in the development of this method, in particular the initial phase and the adaptation of the search neighborhood based on the local structure of a given attraction basin, leading to a version which we term "Reactive Affine Shaker" (RASH) for reasons explained later. We present experimental results on a widely used set of benchmark functions.

This paper is structured as follows. In Section 2 the RASH technique is described and motivated. In Section 3 some crucial aspects of the technique are mathematically analyzed. In Section 4 the experimental results are shown, together with the comparison with other published algorithms.

2 The Reactive Affine Shaker Algorithm

The Reactive Affine Shaker algorithm (or RASH for short) is an adaptive random search algorithm based on function evaluations. The seminal idea of the scheme was presented for a specific application in neural computation in [1]. The current work presents a detailed analysis of the specific form of search executed (called "double shot"), and it proposes a more effective strategy during the initial part of

the search by analyzing the evolution of the search direction in the first iterations, when the search succeeds with probability close to one.

2.1 Motivation and Analysis

The algorithm starts by choosing at random, in the absence of prior knowledge, an initial point $\boldsymbol{x}$ in the configuration space. This point is surrounded by an initial *search region* $\mathcal{R}$ where the next point along the trajectory is searched for.

In order to keep a low computation overhead, the *search region* is identified by n vectors, $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^n$ which define a "box" around the point $\boldsymbol{x}$:

$$\mathcal{R} = \left\{ \boldsymbol{x} + \sum_{i=1}^{n} \alpha_i \boldsymbol{b}_i, \quad \alpha_1, \ldots, \alpha_n \in [-1, 1] \right\}.$$

The search occurs by generating points in a stochastic manner, with a uniform probability in the search region. For a reason which will become clear in the following description, a single displacement $\boldsymbol{\Delta}$ is generated and two specular points $\boldsymbol{x}^{(t)} + \boldsymbol{\Delta}$ and $\boldsymbol{x}^{(t)} - \boldsymbol{\Delta}$ are considered in the region for evaluation (*double shot*, see also [3]). An evaluation is "successful" if the f value at at least one of the two trial points is better than the value at the current point.

By design, RASH is *an aggressive local minima searcher*: it aims at converging rapidly to the local minimizer in the attraction basin where the initial point falls.

We assume that most computational effort during the search is spent by calculating function values $f(\boldsymbol{x})$ at tentative points. Because of the algorithm simplicity, the assumption is valid for non-trivial real-world problems.

The search speed is related to the average size of the steps $\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\|$ executed along the search trajectory. Let's consider two extreme cases. If the search region is very small and the function is smooth, the "double shot" strategy will produce a new successful point with probability close to one, see Section 3.1, but the step will be very small. *Vice versa*, if the search region is very large and it coincides with the initial range of interest, the search strategy will become that of naïf random search: points are generated at random in the entire search space. The step can be large, but the locality assumption is lost and, unless the problem is very simple, a potentially very large number of points will have to be evaluated before finding a successful one. Ideally, to maximize the usage of the information derived from the costly $f(\boldsymbol{x})$ computations, one should aim at the *largest possible step per function evaluation*. This optimal criterion cannot in general be fulfilled, in particular if the analytic form of the function is not known and values $f(\boldsymbol{x})$ are obtained by simulation.

RASH aims at maintaining the search region size as large as possible, while still ensuring that the probability of a success per evaluation will be reasonably close to one. Success probabilities in the range 0.3 - 0.5 are considered acceptable. Now, the success probability is related both to the area of the search region, and to its form. For example, if the attractor basin consists of an elongated and narrow valley leading to a local minimizer, for a fixed area, a search region

elongated along the bottom of the valley will guarantee a higher success rate of the double shot strategy, and therefore longer average step sizes.

RASH obtains both design objectives: (i) success probability per sample close to one and (ii) largest possible step size per successful sample, through a "reactive" determination of the search area during the search. For objective (i) the area is enlarged if the search is successful, reduced if unsuccessful, for objective (ii) the area is elongated along the last successful direction. Of course, "largest possible" has a heuristic meaning: given the partial knowledge about f and the lack of constraints about its functional form we are satisfied if a reasonably large step is determined by a simple reactive scheme.

With more detail, the algorithm proceeds by iterating the following steps:

1. A new tentative point is generated by sampling the local search region $\mathcal{R}$ with a uniform probability distribution and by using the "double shot" strategy. The second specular shot is only evaluated if the first one does not succeed.
2. The search region is modified according to outcome of the tentative point. It is compressed if the new function value is greater than the current one (unsuccessful sample), it is expanded otherwise (successful sample). The region is modified by taking into account the direction of the last tentative step. In RASH, the search area defined by vectors $\boldsymbol{b}_i$ undergoes an affine transformation, see equations (1) and (2) below.
3. If the sample is successful, the new point becomes the current point, and the search region $\mathcal{R}$ is translated so that it becomes centered around the new point.

A last design decision concerns the initial size of the search area, in the absence of initial information about the local attraction basin of f. Two simple options, which do not require critical parameters to be tuned, are to start with a search area corresponding to the initial search range, which will be rapidly compressed in the following iterations until it leads to a success, or, on the contrary, to start with a very small search area, which will be rapidly expanded. The first option is in conflict with the requirement that RASH should scout for the local minimizer corresponding to the basin of attraction of the initial point. If arbitrarily large jumps are permitted at the beginning, all attraction basins could be reachable, with a probability depending on their sizes. Therefore we adopted the second option.

As it is demonstrated in Section 3.1, when the function is smooth and the search region area goes to zero, the probability of success of the "double shot" strategy tends to one, no matter what the initial direction is. This fact creates an undesired effect: after picking the first tentative direction, one will have an uninterrupted sequence of successes. At each step, the search area will be expanded along the last direction, which in turn will be generated with uniform probability in an already elongated region. Through this self-reinforcing mechanism one may easily get an extremely elongated search region, where the elongation tends to be collinear with the first *random* direction, with no influence from the form of the current basin. To avoid this spurious effect, the expansion of the search

region is isotropic in the initial part of the search, until the first unsuccessful direction is encountered, i.e., all box vectors are expanded by the same factor.

The details about the evolution of directions during the initial phase of the search, as well as the experiments related to the correlation between initial search directions, are explained in Section 3.2.

After explaining the design choices, let's now comment on the name (Reactive Affine Shaker). The solver's movements try to minimize the number of jumps towards the minimum region, and this is achieved by constantly changing the movement direction and size. Search region and therefore step adjustments are implemented by a feedback loop guided by the evolution of the search itself, therefore implementing a "reactive" self-tuning mechanism. The constant change in step size and direction creates a "shaky" trajectory, with abrupt leaps and turns. Last, modifications of the search parameters are through an affine transformation on the shape of the search region.

2.2 RASH Pseudo-code

Details of the RASH algorithm are shown in Figure 1. At every iteration, a displacement $\boldsymbol{\Delta}$ is generated so that the point $\boldsymbol{x} + \Delta$ is uniformly distributed in the local search region $\mathcal{R}$ (line 4). To this end, the basis vectors are multiplied by different random numbers in the real range $[-1, 1]$ and added:

$$\boldsymbol{\Delta} = \sum_j \text{Rand}(-1, 1)\boldsymbol{b}_j.$$

Rand$(-1, 1)$ represents a call of the random-number generator. If one of the two points $\boldsymbol{x} + \Delta$ or $\boldsymbol{x} - \Delta$ improves the function value, then it is chosen as the next point. Let us call $\boldsymbol{x}'$ the improving point. In order to enlarge the box along the promising direction, the box vectors $\boldsymbol{b}_i$ are modified as follows.
The direction of improvement is $\boldsymbol{\Delta}$. Let us call $\boldsymbol{\Delta}'$ the corresponding vector normalized to unit length:

$$\boldsymbol{\Delta}' = \frac{\boldsymbol{\Delta}}{\|\boldsymbol{\Delta}\|}.$$

Then the projection of vector $\boldsymbol{b}_i$ along the direction of $\boldsymbol{\Delta}$ is

$$\boldsymbol{b}_i|_{\boldsymbol{\Delta}} = \boldsymbol{\Delta}'(\boldsymbol{\Delta}' \cdot \boldsymbol{b}_i) = \boldsymbol{\Delta}'\boldsymbol{\Delta}'^T\boldsymbol{b}_i.$$

To obtain the desired effect, this component is enlarged by a coefficient $\rho > 1$, so the expression for the new vector $\boldsymbol{b}_i'$ is

$$\begin{aligned} \boldsymbol{b}_i' &= \boldsymbol{b}_i + (\rho - 1)\boldsymbol{b}_i|_{\boldsymbol{\Delta}} \\ &= \boldsymbol{b}_i + (\rho - 1)\boldsymbol{\Delta}'\boldsymbol{\Delta}'^T\boldsymbol{b}_i \\ &= \boldsymbol{b}_i + (\rho - 1)\frac{\boldsymbol{\Delta}\boldsymbol{\Delta}^T}{\|\boldsymbol{\Delta}\|^2}\boldsymbol{b}_i \\ &= P\boldsymbol{b}_i \end{aligned} \tag{1}$$

Variable	Scope	Meaning
f	(input)	Function to minimize
$\boldsymbol{x}$	(input)	Initial point
$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d$	(input)	Vectors defining search region $\mathcal{R}$ around $\boldsymbol{x}$
ρ	(input)	Box expansion factor
t	(internal)	Iteration counter
$\mathbf{P}$	(internal)	Transformation matrix
$\boldsymbol{x}$, $\boldsymbol{\Delta}$	(internal)	Current position, current displacement

```
1.  function AffineShaker (f, x, ( b_j), ρ)
2.      t ← 0;
3.      repeat
4.          Δ ← Σ_j Rand(−1, 1) b_j;
5.          if f( x + Δ) < f( x)
6.              x ← x + Δ;
7.              P ← I + (ρ − 1) ΔΔ^T / ‖ Δ ‖^2;
8.          else if f( x - Δ) < f( x)
9.              x ← x - Δ;
10.             P ← I + (ρ − 1) ΔΔ^T / ‖ Δ ‖^2;
11.         else
12.             P ← I + (ρ^−1 − 1) ΔΔ^T / ‖ Δ ‖^2;
13.         ∀j b_j ← P b_j;
14.         t ← t+1
15.     until convergence criterion;
16.     return x;
```

Fig. 1. The Affine Shaker algorithm

where

$$P = \mathbf{I} + (\rho - 1)\frac{\boldsymbol{\Delta}\boldsymbol{\Delta}^T}{\|\boldsymbol{\Delta}\|^2}. \tag{2}$$

The fact of testing the function improvement on both $\boldsymbol{x} + \boldsymbol{\Delta}$ and $\boldsymbol{x} - \boldsymbol{\Delta}$ is called *double-shot strategy*: if the first sample $\boldsymbol{x} + \boldsymbol{\Delta}$ is not successful, the specular point $\boldsymbol{x} - \boldsymbol{\Delta}$ is considered. This choice drastically reduces the probability of generating two consecutive unsuccessful samples. The motivation is clear if one considers differentiable functions and small displacements: in this case the directional derivative along the displacement is proportional to the scalar product between displacement and gradient $\boldsymbol{\Delta} \cdot \nabla f$. If the first is positive, a change of sign will trivially cause a negative value, and therefore a decrease in f for a sufficiently small step size. The empirical validity for general functions, not

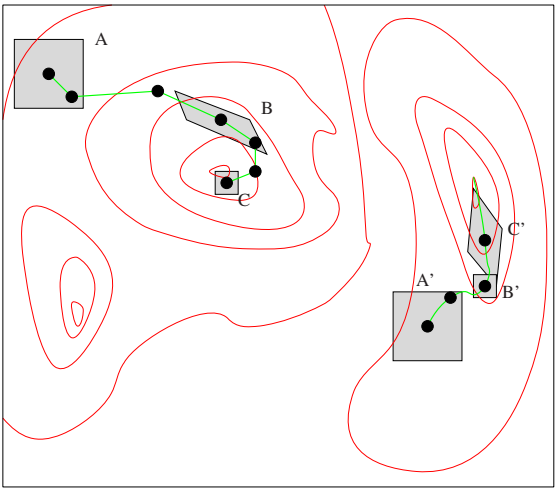


Fig. 2. Affine Shaker geometry: two search trajectories leading to two different local minima. The evolution of the search regions is also illustrated.

necessarily differentiable, is caused by the correlations and structure contained in most of the functions corresponding to real-world problems. See Section 3.1 for a thorough analysis motivating the double-shot strategy.

If the double-shot strategy fails, then the affine transformation (1) is applied by replacing the expansion factor ρ with its inverse ρ^{-1} (line 12 of Figure 1), causing a compression of the search area.

An illustration of the geometry of the Reactive Affine Shaker algorithm is shown in Figure 2, where the function to be minimized (in this case the domain is a square in $n = 2$ dimensions) is represented by a contour plot showing *isolines* at fixed values of f, and two trajectories (ABC and A'B'C') are plotted. The search regions are shown for some points along the search trajectory. A couple of independent vectors define the search region as a parallelogram centered on the point. The design criteria are given by an *aggressive search for local minima*: the search speed is increased when steps are successful (points A and A' in Figure 2), reduced only if no better point is found after the double shot. When a point is close to a local minimum, the repeated reduction of the search frame produces a very fast convergence of the search (point C in Figure 2). Note that another cause of reduction for the search region can be a narrow descent path (a "canyon", such as in point B' of Figure 2), where only a small subset of all possible directions improves the function value. However, once an improvement is found, the search region grows in the promising direction, causing a faster movement along that direction.

2.3 Termination and Repeated Runs

For most continuous optimization problems, an effective estimation of the number of steps required for identifying a global minimum is clearly impossible.

Variable	Scope	Meaning
f	(input)	Function to minimize
ρ,	(input)	Box expansion factor
$L_1, \ldots, L_d, U_1, \ldots, U_d$	(input)	Search range
$L'_1, \ldots, L'_d, U'_1, \ldots, U'_d$	(input)	Initialization range
$\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d$	(internal)	Vectors defining search region $\mathcal{R}$ around $\boldsymbol{x}$
$\boldsymbol{x}$, $\boldsymbol{x}'$	(internal)	Current position, final position of run

1. **function** ParallelAffineShaker (f, ρ, (L'_j), (U'_j), (L_j), (U_j))
2. $\quad \forall j \ \boldsymbol{b}_j \leftarrow \dfrac{U_j - L_j}{4} \cdot \boldsymbol{e}_j$;
3. $\quad$ **pardo**
4. $\quad\quad \boldsymbol{x} \leftarrow$ random point $\in [L'_1, U'_1] \times \cdots \times [L'_d, U'_d]$;
5. $\quad\quad \boldsymbol{x}' \leftarrow$ AffineShaker(f, $\boldsymbol{x}$, ($\boldsymbol{b}_j$), ρ);
6. $\quad$ **return** best position found;

Fig. 3. The Repeated RASH algorithm

Even when a local minimum is located, it is generally impossible to determine whether it is the global one or not, in particular if the knowledge about the function derives only from evaluations of $f(x)$ at selected points.

Because RASH does not include mechanisms to escape from local minima, it should be stopped as soon as the trajectory is sufficiently close to a local minimizer. Suitable termination criteria can be derived, for instance a single RASH run can be terminated if the search region becomes smaller than a threshold value. In fact, the box tends to reduce its volume in proximity of a local minimum because of repeated failures in improving the function value.

By design, RASH searches for local minimizers and is stopped as soon as one is found. A simple way to continue the search after a minimizer is found is to restart from a different initial random point. This approach is equivalent to a "population" of RASH searchers where each member of the population is *independent*, completely unaware of what other members are doing, see on Figure 3. The parallel execution of RASH optimizers is considered in the experimental Section.

3 Analysis and Motivation of the Design Choices

As explained in Section 2, the evolution of the size and the shape of the search region should lead to a ratio of successes in the double-shot strategy comparable to 50%. In this manner both successes and failures will occur and the search area will adapt to the local structure of the function f.

A case in which this assumption is wrong is during the initial phase of the search, when the search region is very small. In fact, Section 3.1 proves that, under reasonable assumptions, the success probability of the double-shot strategy tends to 1 as the size of the search region is reduced.

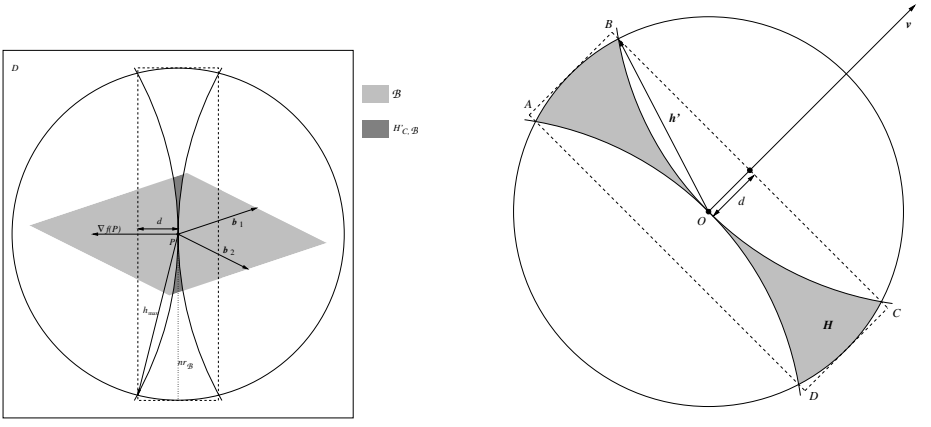


Fig. 4. Left: description of the settings of Theorem 1. Note that $H_{\mathcal{B}}$ is a subset of $H'_{C,\mathcal{B}}$. Right: Description of the setting of Lemmata 1 and 2.

An important issue in the RASH strategy, strongly related to a very high double-shot success ratio, is the dependency between the direction of the initial step (which is random and with high probability of success) and the direction of subsequent steps. The average angle between two random directions in $\mathbb{R}^n$ is exactly determined in Section 3.2. In order to experimentally verify such dependency, the relation shall be used in Section 4.2 to discuss experimental data about this "memory effect".

The final conclusion of this analysis leads to the choice of isotropic expansions ($\boldsymbol{b}'_i = \rho \boldsymbol{b}_i$ for all i) of the search region in the initial phase until the first failure is encountered and the normal affine transformation process (1) starts.

3.1 Double-Shot Success Probability

If the function f is linear, an increase of the function value at the displaced point $\boldsymbol{x} + \boldsymbol{\Delta}$ implies a decrease of the value at the specular point $\boldsymbol{x} - \boldsymbol{\Delta}$, and therefore the "double shot" strategy is trivially bound to be successful. The intuition supporting this strategy for a general function is that, if the function f is smooth, it can be approximated around a given point by a tangent hyperplane with a good accuracy for small displacements. The "double shot" should therefore be successful with a high probability if the search region, and therefore the displacement, becomes very small. The purpose of this section is to analyze in detail the success probability of the strategy used in RASH.

Without loss of generality, suppose that the current point is 0. Let $\mathcal{B}$ be the current box, shown in light grey in the left side of Figure 4:

$$\mathcal{B} = \left\{ \sum_{i=1}^{n} \alpha_i \boldsymbol{b}_i, \quad \alpha_1, \ldots, \alpha_n \in [-1, 1] \right\}.$$

Let us define as $r_{\mathcal{B}} = \max\{\|\boldsymbol{b}_1\|, \ldots, \|\boldsymbol{b}_n\|\}$ the *radius* of the box. Of course, the box $\mathcal{B}$ is contained in the circle with radius $n \cdot r_{\mathcal{B}}$.

Let $H_{\mathcal{B}}$ be the subset of $\mathcal{B}$ where the double shot strategy does not succeed:

$$H_{\mathcal{B}} = \{\boldsymbol{h} \in \mathcal{B} : f(P + \boldsymbol{h}) \geq f(P) \wedge f(P - \boldsymbol{h}) \geq f(P)\}. \tag{3}$$

In Figure 4 (left side) the set $H_{\mathcal{B}}$ is contained in the dark grey area, whose exact meaning shall be made clear later. We want to show that as the box becomes smaller and smaller, the probability of failure of the double shot strategy tends to zero. Since the choice of the vector $\boldsymbol{h} \in \mathcal{B}$ is uniform, we just need to show that the ratio between the measure of $H_{\mathcal{B}}$ and the measure of $\mathcal{B}$ tends to zero, where by *measure* we mean the ordinary (Lebesgue) measure in $\mathbb{R}^n$.

More formally, we want to prove the following.

Theorem 1. *Let $D \subseteq \mathbb{R}^n$, a point $P \in D$, a function $f : D \to \mathbb{R}$ continuous in P with continuous first partial derivatives, a real constant $K > 0$.*

Then, for every $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$, there exists $\delta \in \mathbb{R}$ such that, for every set of vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^n$, defining the box $\mathcal{B}$ (where $P + \mathcal{B} \subseteq D$) with $r_{\mathcal{B}} < \delta$ and $\text{measure}(\mathcal{B}) \geq K r_{\mathcal{B}}^n$*, we have*

$$\frac{\text{measure}(H_{\mathcal{B}})}{\text{measure}(\mathcal{B})} < \varepsilon.$$

Note that, in addition to function regularity hypotheses, a constraint on the measure of $\mathcal{B}$ to be greater of $K r_{\mathcal{B}}^n$ for some constant $K > 0$ has been introduced; this is necessary in order to avoid degenerate cases where two vectors tend to be arbitrarily aligned.

In order to prove Theorem 1 we need the following lemma (see the right side of Figure 4 for a visual representation):

Lemma 1. *Let $C > 0$, $\boldsymbol{v} \in \mathbb{R}^n$ and $H = \{\boldsymbol{h} : \|\boldsymbol{h}\| \leq 1 \wedge |\boldsymbol{h} \cdot \boldsymbol{v}| \leq \|\boldsymbol{h}\|^2\}$. If $\boldsymbol{h}' \in \mathbb{R}^n$ is such that $\|\boldsymbol{h}'\| = 1$ and $\boldsymbol{h}' \cdot \boldsymbol{v} = C\|\boldsymbol{h}\|^2 = C$, then H lies outside the cone generated by rotating $\boldsymbol{h}'$ around $\boldsymbol{v}$.*

In other words, all vectors in H are "more perpendicular" than $\boldsymbol{h}'$ with respect to $\boldsymbol{v}$.

Proof. We just need to show that for every $\boldsymbol{h} \in H$ the normalized projection of $\boldsymbol{h}$ on $\boldsymbol{v}$, i.e., the cosine of their angle, is smaller than that between $\boldsymbol{h}'$ and $\boldsymbol{v}$. Indeed,

$$\frac{|\boldsymbol{h} \cdot \boldsymbol{v}|}{\|\boldsymbol{h}\|\|\boldsymbol{v}\|} \leq \frac{C\|\boldsymbol{h}\|^2}{\|\boldsymbol{h}\|\|\boldsymbol{v}\|} = \frac{C\|\boldsymbol{h}\|}{\|\boldsymbol{v}\|} \leq \frac{C}{\|\boldsymbol{v}\|} = \frac{\boldsymbol{h}' \cdot \boldsymbol{v}}{\|\boldsymbol{h}'\|\|\boldsymbol{v}\|}.$$

This leads to the following corollary (again, see the right side of Figure 4 for a visual representation):

Lemma 2. *Let $C > 0$, $\boldsymbol{v} \in \mathbb{R}^n$, let $H = \{\boldsymbol{h} : \|\boldsymbol{h}\| \leq 1 \wedge |\boldsymbol{h} \cdot \boldsymbol{v}| \leq \|\boldsymbol{h}\|^2\}$. If $\boldsymbol{h}' \in \mathbb{R}^n$ is such that $\|\boldsymbol{h}'\| = 1$ and $|\boldsymbol{h}' \cdot \boldsymbol{v}| = C\|\boldsymbol{h}\|^2 = C$, then H lies in the*

n-dimensional cylinder centered in the origin, with axis along the direction of vector $\boldsymbol{v}$*, having height* $2d$*, where*

$$d = \frac{|\boldsymbol{h}' \cdot \boldsymbol{v}|}{\|\boldsymbol{v}\|}$$

and $(n-1)$*-dimensional base of radius* 1*.*

Proof. Such cylinder is the set of vectors $\boldsymbol{w}$ such that the projection of vector $\boldsymbol{w}$ along the direction of $\boldsymbol{v}$ is less than d, and the norm of the component of $\boldsymbol{w}$ perpendicular to $\boldsymbol{v}$ is less than 1:

$$X_{\boldsymbol{v},d} = \left\{ \boldsymbol{w} \in \mathbb{R}^n : \frac{|\boldsymbol{w} \cdot \boldsymbol{v}|}{\|\boldsymbol{v}\|} \leq d \wedge \|\boldsymbol{w}\|^2 - \left(\frac{\boldsymbol{w} \cdot \boldsymbol{v}}{\|\boldsymbol{v}\|} \right)^2 \leq 1 \right\}.$$

Both conditions are satisfied for all $\boldsymbol{w} \in H$. In fact, by Lemma 1,

$$\boldsymbol{w} \in H \quad \Rightarrow \quad \frac{|\boldsymbol{w} \cdot \boldsymbol{v}|}{\|\boldsymbol{v}\|} \leq \frac{|\boldsymbol{h} \cdot \boldsymbol{v}|}{\|\boldsymbol{h}\|\|\boldsymbol{v}\|} \leq \frac{|\boldsymbol{h}' \cdot \boldsymbol{v}|}{\|\boldsymbol{h}'\|\|\boldsymbol{v}\|} = \frac{|\boldsymbol{h}' \cdot \boldsymbol{v}|}{\|\boldsymbol{v}\|} = d$$

and

$$\boldsymbol{w} \in H \quad \Rightarrow \quad \|\boldsymbol{w}\| \leq 1 \quad \Rightarrow \quad \|\boldsymbol{w}\|^2 - \left(\frac{\boldsymbol{w} \cdot \boldsymbol{v}}{\|\boldsymbol{v}\|} \right)^2 \leq 1.$$

The last lemma enables us to find a convenient upper bound on the measure of the set $H_{\mathcal{B}}$.

Proof (Proof of Theorem 7). Since f has continuous first derivatives, we have

$$f(P + \boldsymbol{h}) = f(P) + \boldsymbol{h} \cdot \nabla f(P) + \sigma(\boldsymbol{h}),$$
$$f(P - \boldsymbol{h}) = f(P) - \boldsymbol{h} \cdot \nabla f(P) + \sigma(-\boldsymbol{h}),$$

where

$$\sigma(\boldsymbol{h}) = O(\|\boldsymbol{h}\|^2). \tag{4}$$

Let $\sigma'(\boldsymbol{h}) = \max\{\sigma(\boldsymbol{h}), \sigma(-\boldsymbol{h})\}$. Then (3) implies:

$$H_{\mathcal{B}} \subseteq H'_{\mathcal{B}} = \{\boldsymbol{h} \in \mathcal{B} : |\boldsymbol{h} \cdot \nabla f(P)| \leq \sigma'(\boldsymbol{h})\}. \tag{5}$$

Equation (4) is still valid for σ', therefore we can find constants C and r_0 such that $\sigma'(\boldsymbol{h}) \leq C\|\boldsymbol{h}\|^2$ as soon as $\|\boldsymbol{h}\| \leq r_0$. Consequently,

$$\forall \mathcal{B}, r_{\mathcal{B}} \leq r_0, \qquad H_{\mathcal{B}} \subseteq H'_{C,\mathcal{B}} = \left\{\boldsymbol{h} \in \mathcal{B} : |\boldsymbol{h} \cdot \nabla f(P)| \leq C\|\boldsymbol{h}\|^2\right\}.$$

The left side of Figure 4 shows the set $H'_{C,\mathcal{B}}$ in dark grey.

Given a box $\mathcal{B}$ having $r_{\mathcal{B}} \leq r_0$ and constrained by the theorem's hypothesis, let us choose a vector $\boldsymbol{h}_{\max}$ such that $\|\boldsymbol{h}_{\max}\| = nr_{\mathcal{B}}$ (so that its "tip" lies on

the sphere) and $\boldsymbol{h}_{\max} \cdot \nabla f(P) = C(nr_{\mathcal{B}})^2$ (so that it lies at the border of the set $H'_{C,\mathcal{B}}$).

As proved in Lemma 2, $H'_{C,\mathcal{B}}$ is contained in the n-dimensional cylinder $P + nr_{\mathcal{B}} \cdot X_{\nabla f(P),\frac{d}{nr_{\mathcal{B}}}}$, i.e. centered in P, with axis directed as $\nabla f(P)$ having base radius $nr_{\mathcal{B}}$ and height $2d$, where

$$d = \frac{\boldsymbol{h}_{\max} \cdot \nabla f(P)}{\|\nabla f(P)\|} = \frac{C(nr_{\mathcal{B}})^2}{\|\nabla f(P)\|}$$

is the projection of $\boldsymbol{h}_{\max}$ along the direction of $\nabla f(P)$. Consequently, whenever $r_{\mathcal{B}} \leq r_0$, as $H_{\mathcal{B}} \subseteq H'_{C,\mathcal{B}} \subseteq P + nr_{\mathcal{B}} \cdot X_{\nabla f(P),\frac{d}{nr_{\mathcal{B}}}}$,

$$\text{measure}(H_{\mathcal{B}}) \leq M(nr_{\mathcal{B}})^{n-1} \cdot 2d = \frac{2MCn^{n+1}}{\|\nabla f(P)\|} r_{\mathcal{B}}^{n+1},$$

where M is the measure of the $(n-1)$-dimensional sphere with unit radius. It follows that

$$\frac{\text{measure}(H_{\mathcal{B}})}{\text{measure}(\mathcal{B})} \leq \frac{2MCn^{n+1}}{\|\nabla f(P)\|} r_{\mathcal{B}}^{n+1} \cdot \frac{1}{Kr_{\mathcal{B}}^n} = \frac{2MCn^{n+1}}{K\|\nabla f(P)\|} r_{\mathcal{B}},$$

therefore, given $\varepsilon > 0$, it is sufficient to let

$$\delta = \min\left\{r_0, \frac{K\|\nabla f(P)\|\varepsilon}{2MCn^{n+1}}\right\}$$

to obtain the thesis.

3.2 Angle between Random Directions in $\mathbb{R}^n$

As we mentioned in Section 2, if RASH starts with a very small and isotropic search region and enjoys an uninterrupted sequence of successes afterwards (caused by the fact that the search region is very small and not by the fact that the chosen directions are suited to the local attraction basin), the average direction obtained after some steps can be very different from a random direction as it can "remember" the initial step. In fact, the affine transformation will elongate the region along the first direction, and therefore the second directions will tend to be approximately collinear with the first one, an effect that will continue for the future iterations. In order to quantify this initial "memory effect", it is of interest to compare the probability distribution of directions obtained after a sequence of affine expansions with a uniform probability.

The problem we are addressing is therefore the following one:

Problem 1. If we draw two random lines in $\mathbb{R}^n$ intersecting at the origin (e.g., by placing two random points on the unit sphere and connecting them to the center), what is the average angle between them?

The problem can also be stated as follows.

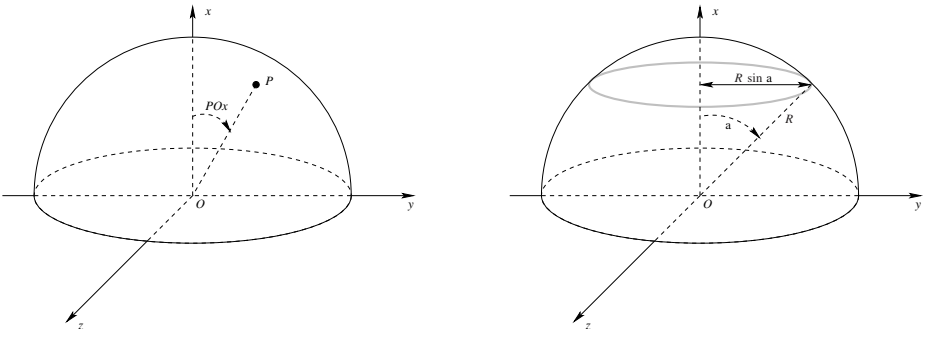


Fig. 5. Left: Angle $\widehat{POx}$ between a random point in the positive-x hemisurface and the positive x axis. Right: The ring whose integration provides the hemisurface.

Problem 2. Given an n-dimensional hypersphere, consider the positive-x hemisphere (the case $n = 3$ is shown in the left side of Figure 5). Choose a random point P on the surface, i.e., a point $P = (x, y, z, \dots) \in \mathbb{R}^n$ such that $x \geq 0$ and $\|P\| = 1$. What is the average value for the angle $\widehat{POx}$?

An inductive expression for the hemisurface

Let $S_n(R)$ be the value of the hemisurface (half the surface of the sphere) for a given number n of dimensions and a given radius R of the n-dimensional hypersphere.

Then, the hemisurface of the $(n+1)$-dimensional hypersphere of radius R can be obtained by integrating the grey ring surface of the right side of Figure 5 for α moving from 0 (the upper point) to $\pi/2$ (the equator):

$$S_{n+1}(R) = \int_0^{\frac{\pi}{2}} 2S_n(R\sin\alpha)R\,\mathrm{d}\alpha. \tag{6}$$

Consider in fact that the grey ring has radius $R\sin\alpha$, and thus its perimeter is $2S_n(R\sin\alpha)$ (twice the hemiperimeter) and its "width" — in the $(n+1)$-th dimension — is equal to $R\,\mathrm{d}\alpha$, since α is expressed in radians.

Notice that the hemisurface of the n-hypersphere is also proportional to the $(n-1)$-th power of R because it is an $(n-1)$-dimensional variety:

$$S_n(R) = C_n R^{n-1} \tag{7}$$

for some positive real constant C_n. This expression shall be useful in the following.

The average angle

Equation (6) is very helpful in calculating the average value of α, which we are looking for. In fact, let $\bar{\alpha}_n$ be the average value of α in n dimensions. Then

$$\bar{\alpha}_n = \frac{\int_S \widehat{POx}\,\mathrm{d}S}{|S|} \tag{8}$$

where S is the hemisurface, the point P scans S and $|S|$ is the measure of S. Consider that angle $\widehat{POx}$ is precisely the angle α of equation (6), and that it is constant within the same ring; then, the probability distribution function of the angle $\widehat{POx}$ is

$$f_n(\alpha) = \frac{2S_{n-1}(R\sin\alpha)R}{\int_0^{\frac{\pi}{2}} 2S_{n-1}(R\sin\alpha)R\,\mathrm{d}\alpha}, \tag{9}$$

and equation (8) becomes

$$\bar{\alpha}_n = \int_0^{\frac{\pi}{2}} \alpha f_n(\alpha)\,\mathrm{d}\alpha = \frac{\int_0^{\frac{\pi}{2}} \alpha \cdot 2S_{n-1}(R\sin\alpha)R\,\mathrm{d}\alpha}{\int_0^{\frac{\pi}{2}} 2S_{n-1}(R\sin\alpha)R\,\mathrm{d}\alpha}.$$

Considering the due simplifications and equation (7), we get

$$\bar{\alpha}_n = \frac{J_{n-2}}{I_{n-2}}, \tag{10}$$

where

$$I_n = \int_0^{\frac{\pi}{2}} \sin^n\alpha\,\mathrm{d}\alpha, \qquad J_n = \int_0^{\frac{\pi}{2}} \alpha\sin^n\alpha\,\mathrm{d}\alpha. \tag{11}$$

With the notation introduced by equations (11), the probability density function (9) can be written as

$$f_n(\alpha) = \frac{\sin^{n-2}\alpha}{I_{n-2}}.$$

Determining I_n and J_n

The values of I_n and J_n can be obtained by straightforward calculations (integration by parts) leading to the following inductive expressions:

$$I_n = \begin{cases} \frac{\pi}{2} & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ \frac{(n-1)I_{n-2}}{n} & \text{if } n \geq 2; \end{cases} \qquad J_n = \begin{cases} \frac{\pi^2}{8} & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ \frac{(n-1)J_{n-2}}{n} + \frac{1}{n^2} & \text{if } n \geq 2. \end{cases} \tag{12}$$

Table 1 reports some representative values of $\bar{\alpha}_n$ in obtained from the analytical expression. Let us note that the average value of the angle tends to become close to 90 degrees as the number of dimensions increases. In other words, two random directions in a high-dimensional space tend to be almost perpendicular.

Table 1. Values of $\bar{\alpha}_n$ for increasing dimensions

n	$\bar{\alpha}_n$ (radians)	$\bar{\alpha}_n$ (degrees)	n	$\bar{\alpha}_n$ (radians)	$\bar{\alpha}_n$ (degrees)
2	0.785398	45.0000	15	1.356416	77.7169
3	1.000000	57.2958	20	1.387008	79.4697
4	1.103708	63.2378	30	1.422228	81.4877
5	1.166667	66.8451	40	1.442770	82.6646
10	1.302778	74.6437	50	1.456625	83.4584

4 Experimental Results

Several tests have been performed in order to both validate and motivate the chosen strategy and to compare it with other state-of-the-art techniques. The proposed RASH algorithm has been tested on various classical test functions, whose analytical formulation and properties are reported in many optimization papers, see for example [4]. After some experiments on the success rate of the double-shot strategy (Section 4.1), a discussion of some spurious effects during the initial phase of the search which are solved by our version (Section 4.2) and an experimental analysis of the technique's robustness to parameter variations (Section 4.3), we introduce the experimental results on the benchmark suite (Section 4.4), and an analysis of the effectiveness of the heuristic when compared with alternative techniques (Section 4.5).

4.1 Success Rate of the Double-Shot Strategy

A measure of the effectiveness of the RASH heuristic can be the double-shot success rate during the search. In Section 3.1 we show that the success rate must be very high at the beginning, when the search region is very small; however, after the initial transient phase, the rate should be reduced because the algorithm dynamically sets a balance between step size and success rate. Figure 6 shows two representative cases. In both plots, the x axis (logarithmic) reports the number of function evaluations, while the y axis reports a simple moving average of the rate of double-shot successes over the previous 100 steps on a representative run. After an initial phase when the success rate is very high due to the small size of the search region (which confirms the analysis of Section 3.1), during a significant portion of the search the double-shot success ratio varies from 30% to 55%.

In the first plot, where a local minimum of a Shekel 4,5 function is reached, we observe a sharp reduction of success rate at the end. This happens when a local minimum is reached, and further improvement becomes impossible. The steep reduction of the double-shot success rate can therefore be used as a restart criterion. In the Rosenbrock case (right plot), the system proceeds slowly towards better values, by following the very narrow valley leading to the global optimum. The success rate remains close to 50% during the descent.

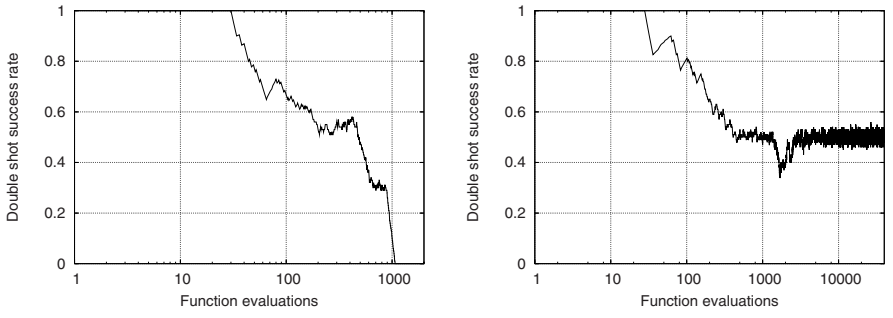


Fig. 6. Success rate of the double-shot strategy for a Shekel 4,5 (left) and a Rosenbrock 10 (right) search

These results confirm the "aggressive" attitude of the RASH heuristic, whose search region $\mathcal{R}$ is continuously adjusted to allow steps as large as possible while still ensuring a large success probability of each double shot trial.

4.2 Influence of Initial Conditions

An issue that deserves further study is the dependence of the search strategy on the initial conditions. In particular, the choice of the first displacement vector $\boldsymbol{\Delta}$ may influence the subsequent behavior in an improper way, because on successful moves the search box will expand along $\boldsymbol{\Delta}$, thus influencing the next choice of $\boldsymbol{\Delta}$, and so on. At the beginning, with a very small search region, the double-shot strategy succeeds with probability close to one, and the local characteristics of the f function do not have a chance of influencing the evolution of the search region in an effective way. The successes depend on the very small size of the box more than on the local properties.

In order to study this effect, we simulated the algorithm's behavior during a sequence of 10 successful applications of the double-shot procedure, by repeatedly generating a $\boldsymbol{\Delta}$ vector in the search box $\mathcal{R}$, then updating $\mathcal{R}$ according to equation (11). Finally, the angle between the directions of the first and the tenth value of $\boldsymbol{\Delta}$ is computed.

Figure 7 shows the average and standard error of 100 runs on different problem dimensions. The continuous plot shows the theoretically calculated average angle between two random directions, as obtained in Section 3.2. It can be seen how a memory effect can be detected after the first iterations. The initial and final directions are correlated, not random. As expected, they tend to be collinear. Because the box elongation can be very large (the search region becomes "needle-like"), a potentially large number of successive iterations can be spent to adapt it to the structure of the local attraction basin.

In order to avoid this spurious effect, as mentioned, in the initial phase of RASH the box is enlarged in an isotropic manner, by multiplying each basis

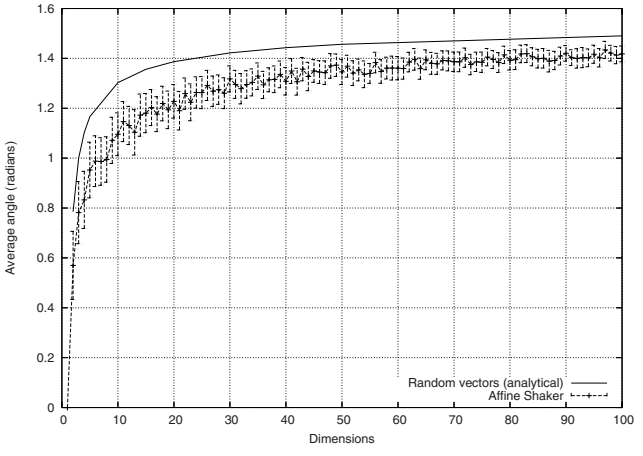


Fig. 7. Angle between the first and the tenth move. The solid line represents the theoretical angle if the two directions were random (see Section 3.2), the dashed series is obtained after 10 subsequent successes of the double shot procedure.

vector $\boldsymbol{b}_i$ by the same ρ factor, until the first lack of success is encountered and the affine transformation (11) is used.

4.3 Robustness w.r.t. Parameter Variations

The RASH technique depends on two parameters: the size of the initial search region and the box expansion factor ρ. The search region is defined by a set of vectors, initially having the same length and orthogonal. Tests have been performed on the statistical dependence of the optimization outcome on the initial length of box vectors. Results of these tests suggest independence, which is easily justified by the observation that the search region adjusts to its proper value after a short transient period, so the overall behavior is affected by different choices in the initial size only at the beginning of its evolution.

Figure 8 reports the outcome of experiments on the dependence of the optimization outcome versus the box expansion factor. Every point in the graph is obtained by averaging 100 runs of 500 function evaluations each, with the same value of ρ, while the error bars represent the 99% confidence interval of the mean. Note that the horizontal axis actually plots the box reduction factor ρ^{-1}, which can be represent more regularly, since its useful values lie in the range $(0, 1)$, where $\rho^{-1} \approx 0$ means a great variability of the search region.

The left plot in Figure 8 shows the average outcome of RASH optimizations of the 5-dimensional Zakharov function, whose global minimum is 0. The right plot shows the equivalent results for the 5-dimensional Shekel function, whose global minimum is below -10.

The two plots report very different behaviors. In particular, all runs locate the minimum of the Zakharov function, which is unimodal, and different values of

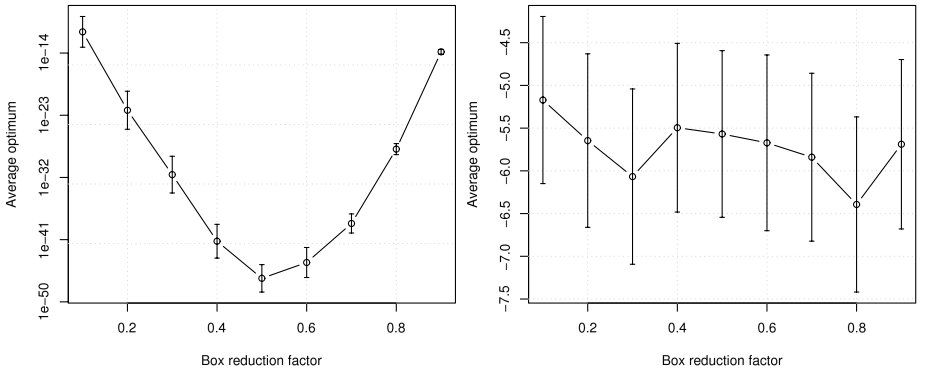


Fig. 8. Robustness against variations of the reduction factor ρ^{-1}: Zakharov 5 (left), Shekel 5 (right)

ρ tend to condition the "fine tuning" to the actual minimum; in this particular case, the optimal value for the reduction parameter is $\rho^{-1} \approx .5$.

The multimodal Shekel function shows a behavior which is far less dependent on the reduction factor. This depends on the fact that many runs do not locate the global minimum, and the chance of falling towards a better minimum is not conditioned by ρ^{-1}.

The conclusion is that the robustness of the RASH heuristic with respect to the value of the search region expansion factor ρ is confirmed for the localization of local minima, while the choice of a correct value is important to improve the precision of the result. In the following, the value $\rho = 2$ shall be used.

4.4 Benchmarks

This section reports the results obtained by running the RASH algorithm on a benchmark suite of various classical test functions, see for example [4] for the definitions.

Every optimization run of RASH begins with a cubic search region defined by an orthogonal set of vectors of length $\|\boldsymbol{b}_i\| = 10^{-4}$. The box expansion factor is $\rho = 2$ and the vectors are expanded isotropically until the first double-shot value fails, after which equation (1) is used as motivated in Section 4.2.

Table 2 shows the results for 100 independent optimization runs for each function. A run is considered successful if the heuristic finds a point x such that

$$f(x) - f_{\min} < \varepsilon_{\text{rel}} |f_{\min}| + \varepsilon_{\text{abs}} \tag{13}$$

where $f_{\min}$ is the known global minimum. Following [4], we have set $\varepsilon_{\text{rel}} = 10^{-4}$ and $\varepsilon_{\text{abs}} = 10^{-6}$. Runs are stopped, and lack of success is recorded, if the global minimum is not located after $5000n$ function evaluations. In the experiments, the termination criteria described in Section 2.3 are disabled in order to evaluate the effectiveness of RASH when applied to the known test cases. The only retained criterion is the maximum number of iterations, while the execution

Table 2. Number of successes, average function evaluations and average minimum found for 100 optimization runs on the test functions. Numbers in parentheses include unsuccessful runs.

f	n	Success	Evals	CPU time	ΔMin
Goldstein-Price	2	76	476 (2762)	0.121 (0.702)	$2.85 \cdot 10^{-3}$
Hartmann d,4	3	96	2227 (2738)	1.73 (2.12)	$4.26 \cdot 10^{-4}$
	6	63	257 (11262)	0.995 (43.6)	$2.24 \cdot 10^{-3}$
Shekel 4,5	4	35	170 (13059)	0.338 (26.0)	0.261
Shekel 4,7	4	31	306 (13895)	0.542 (24.6)	0.370
Shekel 4,10	4	30	164 (14049)	0.362 (31.0)	0.438
Zakharov	10	100	2473	13.9	$9.46 \cdot 10^{-7}$
	20	100	12259	464	$9.86 \cdot 10^{-7}$
	50	100	83605	42099	$9.95 \cdot 10^{-7}$
	100	100	260358	1843765	$9.86 \cdot 10^{-7}$
Rosenbrock	3	100	3595	2.06	$7.98 \cdot 10^{-7}$
	4	65	12085 (14855)	11.0 (13.5)	$9.33 \cdot 10^{-5}$
	5	1	15122 (24901)	28.3 (32.2)	$1.54 \cdot 10^{-3}$

is artificially interrupted, for the experimenters' convenience, when the known global minimum is located with the given degree of accuracy.

The number of successful runs is shown in column *Success*. The average number of function evaluations required in successful runs is shown in column *Evals*, (figures in parentheses include unsuccessful runs). Column *CPU time* reports the average execution time of successful runs (in parentheses, also unsuccessful runs are accounted for), given in standard CPU time units as defined in [6].

Column ΔMin reports the average value of the differences between the minima achieved by 100 runs (including unsuccessful ones) and the actual global minimum.

It can be noted that, for some functions like Goldstein-Price, Hartmann, and Zakharov, the success rate is large and the number of function evaluations is comparable to, and in some cases better than, the number of function evaluations used by more complex techniques like Enhanced Simulated Annealing, see for comparison Table 1 of [13]. These results confirm that the standard behavior of RASH is to rapidly locate a local minimum in the basin of attraction where the initial point lies. However, by design, RASH has no mechanism to escape local minima after they are identified. Therefore it is not surprising that the percentage of success is lower for other functions, like for example for Shekel. While the RASH algorithm shows a good performance on most test functions, a very ill-conditioned problem, such as the Rosenbrock function, is solved in a satisfactory way only for a small number of dimensions. The effects of high dimensionality are also apparent on the *CPU time* column of Table 2. Due to the relative simplicity of the benchmark functions, the dominating factor for

Table 3. Number of successes, average function evaluations and average minimum found for 100 optimization runs on the test functions on $2n$ parallel threads

f	n	Threads	Success	Evals	CPU time	ΔMin
Goldstein-Price	2	4	99	337 (434)	0.152 (.195)	$1.25 \cdot 10^{-4}$
Hartmann d,4	3	6	100	856	0.556	$2.55 \cdot 10^{-4}$
	6	12	100	2420	5.38	$2.41 \cdot 10^{-4}$
Shekel 4,5	4	8	93	1296 (2605)	1.28 (2.57)	$1.2 \cdot 10^{-3}$
Shekel 4,7	4	8	94	1323 (2444)	1.28 (2.36)	$1.03 \cdot 10^{-3}$
Shekel 4,10	4	8	85	1336 (4136)	1.34 (4.15)	$2.53 \cdot 10^{-3}$

a large number of dimensions is the affine transformation of the search region vectors, amounting to n vector multiplications by an $n \times n$ matrix, totaling to $O(n^3)$ time per optimization step. For high-dimensional problems and functions requiring small computation more specialized techniques like ESA of [13] should be considered. In any case, let's note that many functions are extremely costly to compute, see for example evaluations requiring the simulation of an industrial plant or a real-world experimentation.

In order to obtain higher success rates, we exploited the fast convergence speed of the successful runs by *parallelizing* independent solvers on the same function. Considering independent repetitions is in fact the simplest way to use the simple RASH component to obtain a more robust scheme. In this case, an optimization session is achieved by instantiating $2n$ independent solvers, where n is the dimension of the function's domain, and by executing one step of each solver in a round-robin fashion until either one of the solvers finds a value that satisfies equation (13) or the maximum number of function evaluations is reached. The total number of evaluations by all independent threads is counted.

The results of interest, where parallelization actually leads to an improvement, are shown in Table 3. Note that most problem instances benefit from parallel search. However, highly dimensional problems such as Zakharov are already solved with a single thread. In this case, a single solver is more effective, and the success rate for a fixed number of iterations decreases if parallel threads are exploited.

4.5 Comparison with Other Techniques

The RASH algorithm behavior has been compared with other local search heuristics, and results are presented in Table 4. In particular, we focused on two recent proposals, Enhanced Simulated Annealing [13] and Enhanced Continuous Tabu Search [4], which, like RASH, aim at minimizing functions of continuous variables. Another classical proposal, the Improved Simulated Annealing algorithm [15] is shown in two different variants (wide and narrow search domain). Techniques have been selected on the basis of similar hypotheses (continuous functions, no analytical tools other than evaluation) and on similar criteria for estimating success and efficiency.

Table 4. Comparison with other techniques — number of successful minimizations; see text for the description of the techniques

Method	$G.-P.$	H_3	H_6	$S_{4,5}$	$S_{4,7}$	$S_{4,10}$
RASH	99	100	100	93	94	85
ECTS	100	100	100	75	80	75
ESA	100	100	100	54	54	50
ISA 1	n.a.	99	97	7	1	3
ISA 2	n.a.	100	0	19	28	18

Table 5. Comparison with other techniques — number of function evaluations; see text for the description of the techniques

Method	$G.-P.$	H_3	H_6	$S_{4,5}$	$S_{4,7}$	$S_{4,10}$
RASH	434	856	2420	2605	2444	4136
ECTS	231	548	1520	825	910	898
ESA	783	698	1638	1487	1661	1363
ISA 1	n.a.	6965	21802	6030	2936	3562
ISA 2	n.a.	13758	1116	8568	8631	7824

For comparison purposes, we rely on data provided in [4] and on the original sources. The definition of "successful" run takes into account the criteria defined in [4,13], where the maximum number of allowed evaluations is $5000n$, as described before. For RASH, we chose to employ the multi-thread results shown on Table 3. Since the RASH algorithm is targeted at local minimization, while the other techniques implement global mechanisms, comparison of such techniques with the parallel version of RASH is more fair.

The results clearly show that the RASH heuristic achieves state-of-the-art results on various classical problems, ranging from 85% to 100% successful minimizations. This result is of interest because of the simplicity of the technique, which can be an effective building block for more complex heuristic schemes. Table 5, on the other hand, shows that techniques such as ECTS and ESA require less evaluations on the average. When confronted with the number of successful runs, this result suggests that the termination criteria adopted in ECTS and ESA could be actually relaxed in order to achieve a better success ratio, and that a good termination criterion must be devised for the RASH heuristic to become competitive in terms of function evaluations.

5 Conclusions

We proposed and analyzed the Reactive Affine Shaker adaptive random search algorithm based on function evaluations. The main algorithmic contribution of this paper consists of a careful analysis of the double-shot strategy motivating

and quantifying what was originally proposed based on intuition. Furthermore, the paper motivates and analyzes a modified initial phase to avoid a dangerous effect during the initial growth of the search region. Finally, it considers the evaluation of a simple "portfolio" consisting of independent runs of the local RASH searcher started from different random initial configurations.

The conclusions of the experiments show a performance which is in some cases comparable or better w.r.t. competitive techniques. The results are unexpected given the algorithmic simplicity of RASH, in particular its design based on converging rapidly to the local minimizer in the attraction basin of the initial point. The effectiveness is caused by the rapid and effective adaptation of the search region based on feedback from function evaluations at random points. This work motivates the consideration of this component for more complex meta-heuristic schemes.

The algorithm has been designed and implemented as a set of reusable software components to facilitate the experimentation within more advanced schemes. The package is available for evaluation purposes and scientific research at http://www.reactive-search.org/

References

1. Battiti, R., Tecchiolli, G.: Learning with first, second and no derivatives: A case study in high energy physics. Neurocomp. 6, 181–206 (1994)
2. Battiti, R., Tecchiolli, G.: The reactive tabu search. ORSA Journal on Computing 6(2), 126–140 (1994)
3. Brunelli, R., Tecchiolli, G.: On random minimization of functions. Biological Cybernetics 65(6), 501–506 (1991)
4. Chelouah, R., Siarry, P.: Tabu search applied to global optimization. European Journal of Operational Research 123, 256–270 (2000)
5. Corana, A., Marchesi, M., Martini, C., Ridella, S.: Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. ACM Trans. Math. Softw. 13(3), 262–280 (1987)
6. Dixon, L.C.W., Szegő, G.P. (eds.): Towards Global Optimization 2. North-Holland, Amsterdam (1978)
7. Glover, F.W., Kochenberger, G.A.: Handbook of Metaheuristics. International Series in Operations Research and Management Science, vol. 57. Kluwer Academic Publishers, Norwell (2003)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer Academic Publishers, Boston (1989)
9. Gomes, C.P., Selman, B.: Algorithm portfolios. Artif. Intell. 126(1-2), 43–62 (2001)
10. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. J. ACM 8(2), 212–229 (1961)
11. Hoos, H.H., Stützle, T.: Stochastic Local Search Foundations and Applications. Morgan Kaufmann / Elsevier (2004)
12. Pardalos, P.M., Resende, M.G.C. (eds.): Handbook of Applied Optimization. Oxford University Press, NY, USA (2002)
13. Siarry, P., Berthiau, G., Durbin, F., Haussy, J.: Enhanced simulated annealing for globally minimizing functions of many-continuous variables. ACM Transactions on Mathematical Software 23(2), 209–228 (1997)

14. Solis, F.J., Wets, R.J.-B.: Minimization by random search techniques. Mathematics of Operations Research 6(1), 19–30 (1981)
15. Tsoi, A.C., Lim, M.: Improved simulated annealing technique. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Piscataway, NJ (USA), pp. 594–597. IEEE Press, Los Alamitos (1988)
16. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)

Market Based Allocation of Transportation Orders to Vehicles in Adaptive Multi-objective Vehicle Routing

Martin Josef Geiger and Wolf Wenger

Lehrstuhl für Industriebetriebslehre, Universität Hohenheim,
70593 Stuttgart, Germany
mjgeiger@uni-hohenheim.de, w-wenger@uni-hohenheim.de

Summary. The article describes a study on vehicle routing problems under multiple objectives. In particular, we investigate the effectiveness of different approaches when assigning orders to vehicles. The resulting clustering problem is studied within a general framework for multi-objective vehicle routing problems where different vehicle agents place bids for orders which are offered on a marketplace. This marketplace gathers information about the current situation and provides the basis for the resolution of the allocation problem. By implementing different specialized but interacting software agents, an adaptation of the concept to various configurations of the studied problem is possible. Experimental investigations of different assignment logics on benchmark instances have been carried out and numerical results are reported. In brief, a tendency towards a particular clustering approach can be observed.

Keywords: Vehicle routing problem, multi-objective, market based allocation, multi-agent approach.

1 Introduction

The distribution of goods plays an increasingly important role in the modern supply-chain of companies. On the one hand, the cost-efficiency of delivery and storage is considered, minimizing occurring costs throughout the supply chain and as a result creating competitive advantages. On the other hand, the quality of service has to be addressed as more and more companies tend to store the smallest possible amount of goods only, following a just-in-time (JIT) delivery strategy. Consequently, meeting delivery dates becomes increasingly important for the avoidance of out-of-stock situations. When combining both aspects, multi-criteria planning problems are derived which take several optimality criteria simultaneously into consideration [9]. From a scientific management perspective, methods, algorithms and systems are needed to support these complex planning problems. In a practical dynamic environment, this implies that the proposed approaches are flexible, allowing an adaptation to changing situations, constraints, optimality criteria, preferences, etc.

The vehicle routing problem (VRP) is one of the main optimization problems in the context of distribution management, which is faced by numerous

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 119–132, 2008.
springerlink.com

companies and organizations each day. Characteristics of a specific practical problem such as regarded objectives or required constraints are highly variable, and conditions vary from one real world application to the other. However, most of them can be defined on a complete directed network $G = (V(G), A(G))$ with a node set $V(G)$, and a set of arcs $A(G)$ connecting the nodes. In the most classical version with a single depot each node $i \in V(G) \setminus \{0\}$ describes a customer by using various associated parameters, e. g. a corresponding non-negative demand q_i, a non-negative service time d_i or a given time window $[e_i, l_i]$ during which customer i should be or has to be served. Node 0 corresponds to the depot, where a fleet of vehicles with given capacity and/or route length restrictions is stationed to serve the costumers. Analog to the nodes, each arc $(i, j) \in A(G)$ possesses several parameters, foremost a travel-distance a_{ij}, a travel-time t_{ij} or travel-costs c_{ij} occurring by using the connection between i and j.

Several extensions of the classical vehicle routing problem can be found in practice and in literature. Some of them introduce multiple depots, heterogeneous vehicles or the possibility of open routes, where vehicles do not return to the place they depart from. Others take into consideration the dynamics of changing environments [5]. This may result in time-dependent travel times t_{ij} which vary from busy rush hours to quiet times, or in dynamically arriving orders that are either accepted and integrated into the plan or rejected.

Minimizing total travel-distance, total travel-time or overall travel-costs are commonly used objectives in VRPs. More recent approaches aim to identify solutions that provide a high quality of delivery service. In these applications customer satisfaction should be improved, e. g. by integrating specific aspects like the mentioned time windows as hard constraints [29], as soft constraints with some sort of penalty that occurs when a time window is violated [30], or by adding an objective function that minimizes total or maximum tardiness of the served orders [11]. Other service-oriented aspects are balanced workloads of drivers [17, 18] or balanced inequities between the best and the least served costumer [23]. Along with this, VRPs are more and more recognized as multi-objective optimization problems [11, 16, 17, 20, 21, 27]. In this context decision support is not provided by calculating a single optimal solution, but needs to identify a set of Pareto-optimal solutions P and to guide the selection of a most preferred one $x^* \in P$.

Unfortunately, most problems of this domain are $\mathcal{NP}$-hard. Given a fixed fleet size, even finding a feasible solution to the VRP with time windows turns out to be $\mathcal{NP}$-complete itself [28]. As a result, research has concentrated on heuristics and more recently on metaheuristics to obtain good quality solutions in short computing times [14, 25, 26]. Specialized techniques have been used to improve known results for particular VRPs [2] or to provide upper bounds for exact algorithms like column generation algorithms or branch-and-cut algorithms. It has to be mentioned however, that with the increasing specialization of techniques a decrease in generality of the resolution approaches follows. As a result, heuristic optimization frameworks such as HotFrame [10], EasyLocal++ [8],

ParadisEO [3, 19] or the MALLBA library [1] try to address this issue by providing generic libraries for the resolution of optimization problems.

The article is organized as follows. In the following Section 2, a framework for interactive multi-objective vehicle routing is presented that aims to address two critical issues. First, the resolution approach has to be able to solve a range of problems of different characteristics and therefore needs to be of sufficient generality. Second, multiple objectives are integrated and the decision maker is allowed to interact with the system by articulating and adapting individual preferences during the resolution procedure.

An implementation of the framework for multi-objective vehicle routing problems and experimental investigations are presented in Section 3. The system is used to solve instances of multi-depot vehicle routing problems under multiple objectives. In the study presented in this article, we focus in particular on the clustering problem when assigning orders to vehicles. Conclusions are presented in Section 4.

2 A Multi-agent Approach for Interactive Multi-objective Vehicle Routing

Independent from the precise characteristics of the particular VRP, two types of decisions have to be made when solving the problem.

1. The assignment of customers to vehicles (clustering).
2. The construction of a route for a given set of customers (sequencing).

It is well-known that both types of decisions influence each other to a considerable extent. While the clustering of customers to vehicles is an important input for the sequencing, the sequencing itself is of relevance when adding customers to routes as e. g. constraints of maximum travel distances and/or times have to be respected. The two types of decisions can be made either sequential (cluster first-route second vs. route first-cluster second) or in parallel.

The intersection of decisions known from bin-packing (clustering of customers) and the travelling salesman problem (sequencing of customers) results in a problem structure which is considerable more difficult. Even for the most simplistic classical vehicle routing problem known from the scientific literature, obtaining an optimal solution is challenging and quickly becomes infeasible even for medium-sized instances. When considering complex side constraints, this effect is even more present.

As a consequence, we chose to decompose the different types of decisions that have to be made in the VRP, and propose a framework that consists of different agents, each of which addresses a particular aspect of the problem. While each agent is specialized towards optimizing a certain sub-structure, such as minimizing the length of a particular route, an overall solution is obtained by market-based exchange of the gathered information.

Figure 1 gives an overview about the elements of the framework [12].

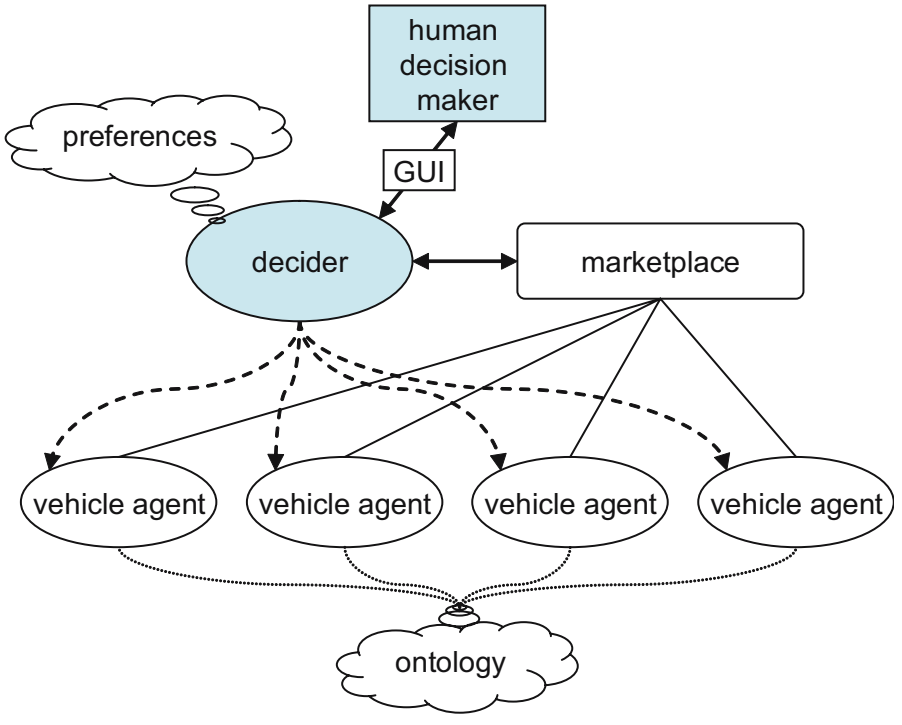


Fig. 1. Illustration of the framework for interactive multi-objective vehicle routing

- The *marketplace* represents the element where orders are offered for transportation and bids for orders, generated and placed by the vehicles, are gathered. In the initial step of the problem solving procedure, all orders are put on the marketplace. The marketplace does not posses any computational intelligence as such, it only consists of a single agent which gathers information from agents and makes the obtained information available to other agents.
- *Vehicle agents* represent the vehicles stationed at the depots. They therefore store information about the current position of the vehicle, its technical details (capacity, speed, maximum distance, etc.), and its currently assigned orders and route. While at the beginning of the optimization procedure the set of assigned orders is empty, routes are subsequently constructed throughout the optimization procedure.

  During the problem resolution process, the vehicle agents place bids for the orders on the *marketplace*. These bids take into consideration the current routes of the vehicles and the potential change when integrating an additional order. Integrating additional orders into existing routes leads to an increase in terms of travelled routes and/or time window violations. This information is reported back to the marketplace where it can be used to compare bids from different vehicles, thus leading to an assignment of orders to vehicles.

In addition to computing bids for orders on the marketplace, the vehicle agents possess computational intelligence techniques to optimize their travelled routes. The actual implementation of these techniques is based on local search, resequencing the orders until a sufficient approximation of the optimal route is identified. With respect to the initially mentioned two types of decision in the VRP, the vehicle agents solve the *sequencing problem*.

- An *ontology* describes the possible properties of the vehicles such as their capacity, availability, current location, technical details, etc. This easily allows the consideration of different types of vehicles (heterogeneous fleet). It also helps to model open routes, where vehicles do not necessarily return to the depot where they depart from.
- A *decider agent* communicates with the human *decision maker* via a graphical user interface (GUI) and builds a preference model of the individual preferences of the human planner. In comparison to generic graphical user interfaces for multi objective optimization such as GUIMOO [4] we chose an approach that also visualizes the actual solution on a map [12], not only the evaluation function value of the currently considered solution. This allows a close investigation of the current solution.

  The decider assigns orders to vehicles, taking into consideration the gathered bids placed for the specific orders. Precisely, this agent represents the *assignment/clustering* logic of orders to vehicles.

A solution is constructed by placing the orders on the marketplace, collecting bids from the vehicle agents, and assigning orders to vehicles while constantly updating the bids. Route construction by the vehicle agents is done in parallel using local search heuristics so that a route can be identified that optimizes the preference model of the decision maker. Reviewing the mentioned variants of solving the clustering/sequencing problems, the presented approach follows the concept of combining both decisions in parallel. The precise method of constructing a solution is given in Algorithm 1.

Algorithm 1. Solution construction procedure

```
1: place orders on the marketplace
2: repeat
3:    for all orders on the marketplace do
4:       gather bids for orders from the vehicle agents
5:    end for
6:    decider agent: select some order from the marketplace
7:    decider agent: select some bid for the selected order
8:    decider agent: assign order to vehicle which generated the bid
9:    vehicle agent: update the route integrating the assigned order
10: until all orders are assigned
```

It is important to mention that in the proposed framework, the decision maker is allowed to change his/her preferences during the construction of the solution. In this case, the *decider agent* updates the stored preference information and in

consequence, the vehicles resequence their orders such that the routes are again optimized for the updated preference information.

3 Implementation and Experiments

3.1 System Configuration

The framework has been implemented in a computer system. In the experiments that have been carried out for this article, two objective functions are considered, the total travelled distances D and the total tardiness T caused by vehicles arriving after the upper bound l_i of the time window. It should be noticed however, that neither the concept presented in Section 2 nor the actual implementation are restricted to the two objective functions only. The general applicability of the framework to other objectives, in particular to the minimization of the maximum tardiness T_{max}, has been shown in [13].

The preferences of the decision maker are represented by introducing a weighted sum of both objective functions. Using the relative importance of the distances $w_D, 0 \leq w_D \leq 1$, the overall utility U of a particular solution can be computed as given in Expression (1).

$$U = w_D \; D + (1 - w_D) \; T \tag{1}$$

In our current implementation, the vehicle agents are able to modify the sequence of their orders using four different local search neighborhoods.

- Inverting the sequence of the orders between positions p_1 and p_2, $p_1 \neq p_2$. While this may be beneficial with respect to the distances, it may pose a problem for the time windows as usually orders are served in the sequence of their time windows.
- Exchanging the positions p_1 and p_2, $p_1 \neq p_2$ of two orders.
- Moving an order from position p_1 and reinserting it at position p_2, $p_1 < p_2$ (forward shift).
- Moving an order from position p_1 and reinserting it at position p_2, $p_1 > p_2$ (backward shift).

In each step of the vehicles' local search procedure, a neighborhood is randomly picked from the above given set of neighborhoods. We select each neighborhood with equal probability of $\frac{1}{4}$. Then, a random neighboring solution is computed based on the chosen neighborhood structure, e. g. by exchanging two randomly chosen jobs or shifting some job to some other position. The so computed move is accepted if an improvement of the route is obtained, always with respect to the particular utility function as given in Expression 1. In brief, the here presented concept implements a reduced Variable Neighborhood Search approach [15]. This concepts has the advantage of overcoming local optimality with comparably little effort as a set of different operators is repeatedly used. Having a whole set of neighborhood operators is particularly beneficial here as the vehicle agents need to be able to construct routes not only minimizing the length

Algorithm 2. Least cost bid generation of the vehicle agents

Require: current route $R_j = \{v_{j1}, \ldots, v_{jn}\}$, order v_i to be integrated in R_j
Set $mincost = \infty$, $R_j^{\min} = \emptyset$
for all insertion points in the current route **do**
  insert order, obtaining modified route R'_j
  if R'_j is feasible **then**
    evaluate new route with respect to the preferences of the decision maker (compute $U(R'_j)$)
    if $U(R'_j) - U(R_j) < mincost$ **then**
      $mincost = U(R'_j) - U(R_j)$
      $R_j^{\min} = R'_j$
    end if
  end if
end for

but also the total tardiness. Simply, a single operator appears less likely to be able to address both aspects at once.

Bids for orders on the marketplace are generated by the vehicle agents, taking into consideration all possible insertion points in the current route. The sum of the weighted increase in distance D and total tardiness T gives the prize for the order. This price reflects the individual preferences articulated by the decision maker using the w_D parameter which expresses the tradeoff between distances and time window violations. The following Algorithm 2 illustrates the computational procedure of the vehicles agents for obtaining the least cost insertion point of a given order.

After all bids have been computed by all vehicle agents and gathered on the marketplace, the decider agent assigns orders to vehicles following a particular logic, e. g. myopically minimizing the cost of the next order assignment.

Initial experiments of the framework on benchmark instances investigated the adaptability of the concept to changing inputs on the decision maker [12]. We have been able to observe that the assignment of customers to routes plays an important role for the later adaptation of the solutions. As a corollary, a deeper investigation of the assignment logic, which we are going to present in the following subsection, appears beneficial.

3.2 Experiments

Two different assignment logics have been investigated for the decider agent. First, orders have been assigned giving priority to the bid with the *minimum price*. Second, the assignment has been done giving priority to the bid with the minimum *alternative cost*. This measure is derived by computing the resulting cost difference when *not* assigning an order to a particular vehicle, and therefore having to assign it to some other vehicle, resulting in the payment of another price. The precise computation of this measure consequently is the minimum difference of all other prices to the particular price of a vehicle. The minimum possible numerical value is 0, which is reached when an alternative vehicle exists

Table 1. Problem characteristics of the instances

Instance	vehicles	orders	depots
pr01	2	48	4
pr02	3	96	4
pr03	4	144	4
pr04	5	192	4
pr05	6	240	4
pr06	7	288	4
pr07	2	72	6
pr08	3	144	6
pr09	4	216	6
pr10	5	288	6

with the same price. A maximum value of ∞ has to be assumed when an order can only be assigned to a single vehicle, making this assignment most critical for the construction of a feasible solution.

The two assignment logics of the optimization framework have been tested on ten benchmark instances taken from [6]. The instances range from 48 to 288 customers that have to be served from 4 to 6 depots, each of which possesses 2 to 7 vehicles vehicles, and may be obtained e. g. from `http://neo.lcc.uma.es/radi-aeb/WebVRP/`. The precise description of the instances is given in [6] and therefore not repeated here. Instead, we give the key characteristics of the instances in the following Table 1.

A full mathematical description of the considered problem has been introduced by [7]. We however modify the precise formulation with respect to the objective functions and consider a multi-objective case of the VRP as given above.

We computed for each assignment logic/problem instance-combination a solution while considering different relative importance values of the distances w_D. The values of w_D varied from 0.1 to 0.9 in steps of 0.1. Extremal values of 0 and 1 have been excluded in the experiments as we did not aim to compute extremal solutions for a single objective function only but focus the investigation to multi-objective vehicle routing. Also, it is important to mention that values of $w_D = 0$ led to difficulties for the minimum price assignment logic where for some instances no feasible solution was found. The algorithm here started to construct routes which initially were optimal with respect to the total tardiness, however traveling routes of such long distances that is later became infeasible for the vehicles to accept additional orders.

3.3 Results

The computational results of the experiments are given in Table 2. As the procedures are deterministic, a single run has been sufficient for each parameter

setting. We give for each instance and value of w_D the rounded values of D, T and the overall utility U, depending on the assignment logic. In column 'Diff.', the relative difference of the utility value obtained by the minimum price assignment logic to the utility value obtained by the alternative cost approach is given. Negative values indicate that *minimum price* led to a better overall result, while positive values state the opposite.

For an easier analysis, column 'Indic.' gives an indicator which illustrates the relative differences between the two approaches in a graphical way. Bullets (•) indicate a better performance of *alternative cost*, and the amount of the symbols categorizes the relative difference in steps of ±5%.

Table 2. Results

		Alternative cost			Minimum price				
Inst.	w_D	D	T	U	D	T	U	Diff.	Indic.
pr01	0.1	1362	0	136	1390	1	140	2.67%	•
	0.2	1305	8	267	1390	1	279	4.11%	•
	0.3	1270	13	390	1390	1	418	6.67%	••
	0.4	1334	21	546	1311	24	539	-1.35%	○
	0.5	1342	41	692	1311	24	667	-3.65%	○
	0.6	1306	115	830	1191	75	744	-11.48%	○○○
	0.7	1169	103	849	1164	75	837	-1.37%	○
	0.8	1169	103	955	1102	154	912	-4.74%	○
	0.9	1118	214	1027	1021	256	944	-8.78%	○○
pr02	0.1	2391	6	244	2385	1	240	-1.87%	○
	0.2	2331	6	471	2385	1	478	1.57%	•
	0.3	2206	28	682	2395	1	719	5.22%	••
	0.4	2169	52	899	2333	46	961	6.49%	••
	0.5	2176	58	1117	2227	116	1172	4.66%	•
	0.6	2100	63	1285	2226	62	1360	5.54%	••
	0.7	2073	78	1475	2160	156	1559	5.42%	••
	0.8	2002	408	1683	1979	162	1616	-4.15%	○
	0.9	1743	506	1619	1801	303	1651	1.93%	•
pr03	0.1	3455	3	349	3689	3	371	6.14%	••
	0.2	3346	27	691	3689	3	740	6.68%	••
	0.3	3336	27	1019	3725	35	1142	10.74%	•••
	0.4	3213	53	1317	3399	52	1391	5.27%	••
	0.5	3152	70	1611	3584	53	1819	11.40%	•••
	0.6	3206	98	1963	3183	164	1975	0.63%	•
	0.7	3034	167	2174	3466	174	2479	12.28%	•••
	0.8	3032	277	2481	2983	167	2420	-2.53%	○
	0.9	2804	372	2561	2793	277	2541	-0.77%	○
pr04	0.1	3970	2	398	4481	0	449	11.19%	•••
	0.2	3975	4	798	4109	19	837	4.58%	•
	0.3	3891	20	1182	4176	34	1277	7.45%	••
	0.4	3644	40	1481	4119	138	1730	14.39%	•••

Table 2. (*continued*)

		Alternative cost			Minimum price				
Inst.	w_D	D	T	U	D	T	U	Diff.	Indic.
	0.5	3533	138	1836	4206	85	2145	14.43%	● ● ●
	0.6	3379	199	2107	3610	181	2239	5.88%	●●
	0.7	3233	213	2327	3795	184	2711	14.17%	● ● ●
	0.8	3059	242	2496	3491	376	2868	12.98%	● ● ●
	0.9	2920	349	2663	3123	356	2846	6.44%	●●
pr05	0.1	4501	6	455	4624	5	467	2.55%	●
	0.2	4500	14	911	4204	8	848	-7.49%	○○
	0.3	4283	21	1300	4116	37	1261	-3.12%	○
	0.4	4451	40	1805	4161	52	1696	-6.42%	○
	0.5	4500	66	2283	4170	86	2128	-7.27%	○○
	0.6	4408	159	2709	3994	107	2439	-11.05%	○ ○ ○
	0.7	4267	112	3020	3835	201	2745	-10.05%	○ ○ ○
	0.8	3898	401	3198	4012	153	3240	1.29%	●
	0.9	3899	482	3557	3494	303	3175	-12.03%	○ ○ ○
pr06	0.1	5540	4	557	5302	1	532	-4.89%	○
	0.2	5297	35	1088	5130	15	1038	-4.79%	○
	0.3	5222	35	1591	5399	26	1638	2.86%	●
	0.4	5184	40	2098	5376	39	2174	3.49%	●
	0.5	4989	68	2529	5298	73	2686	5.83%	●●
	0.6	4866	154	2981	5268	153	3222	7.48%	●●
	0.7	4715	176	3353	4707	212	3358	0.15%	●
	0.8	4330	273	3519	4608	231	3732	5.72%	●●
	0.9	4106	399	3735	4379	434	3985	6.26%	●●
pr07	0.1	1864	0	186	1818	0	182	-2.55%	○
	0.2	1864	0	373	1818	0	364	-2.55%	○
	0.3	1828	8	554	1780	7	539	-2.69%	○
	0.4	1834	32	753	1846	15	747	-0.79%	○
	0.5	1801	32	917	1846	15	930	1.43%	●
	0.6	1737	53	1063	1691	156	1077	1.24%	●
	0.7	1658	143	1204	1576	156	1150	-4.68%	○
	0.8	1619	220	1339	1596	156	1308	-2.37%	○
	0.9	1408	347	1302	1465	194	1338	2.66%	●
pr08	0.1	3375	3	340	3108	0	311	-9.39%	○○
	0.2	3136	17	641	3065	16	626	-2.40%	○
	0.3	3233	30	991	2892	71	917	-7.99%	○○
	0.4	3048	30	1237	2847	83	1189	-4.06%	○
	0.5	2959	119	1539	2950	83	1517	-1.47%	○
	0.6	2808	130	1737	2888	215	1819	4.53%	●
	0.7	2815	233	2041	2494	301	1836	-11.12%	○ ○ ○
	0.8	2619	257	2147	2424	180	1976	-8.66%	○○
	0.9	2239	379	2053	2534	361	2317	11.41%	● ● ●

Table 2. (*continued*)

		Alternative cost			Minimum price				
Inst.	w_D	D	T	U	D	T	U	Diff.	Indic.
pr09	0.1	4148	1	416	4283	1	429	3.03%	•
	0.2	4025	45	841	4108	6	826	-1.76%	○
	0.3	3888	45	1198	4062	41	1247	3.99%	•
	0.4	3817	52	1558	4106	54	1675	6.99%	••
	0.5	3760	56	1908	4116	73	2095	8.92%	••
	0.6	3581	175	2219	3995	79	2429	8.67%	••
	0.7	3620	161	2582	3779	70	2667	3.17%	•
	0.8	3597	280	2934	3672	146	2967	1.12%	•
	0.9	3188	370	2907	3495	226	3168	8.26%	••
pr10	0.1	5508	2	552	5669	1	568	2.72%	•
	0.2	5340	31	1093	5731	2	1148	4.75%	•
	0.3	5360	40	1636	5484	16	1657	1.24%	•
	0.4	5465	80	2234	5523	30	2227	-0.29%	○
	0.5	5146	133	2639	5261	73	2667	1.03%	•
	0.6	4828	271	3005	5318	73	3220	6.67%	••
	0.7	4842	269	3470	4983	276	3571	2.82%	•
	0.8	4516	255	3664	4602	138	3709	1.23%	•
	0.9	4310	353	3914	4593	324	4166	6.04%	••

When analyzing the investigated assignment logics, it becomes clear that differences in the quality of the obtained solutions exist. The *alternative cost* assignment logic led in more cases to superior results compared to the *minimum price* approach. This behavior is especially obvious for instances pr03, pr04, pr09 and pr10. Other instances such as pr02 and pr06 show the same tendency, however with a considerable smaller significance. It is however interesting to see that there are several counterexamples, namely instances pr01, pr05 and pr08 where the opposite conclusion is reached.

There does not appear to be an significant influence of the parameter w_D on the relative performance of the two assignment logics. For all values of w_D, one or the other clustering approach led to superior results, always depending on the particular instance. The recommendation for one or the other assignment logic appears to be based and depending on the instance as such, and not on the relative importance of the criteria.

It has to be pointed out, that several observed differences are rather small. While there are instances in which a particular logic reliably leads to better results, the are some cases in which the differences do not permit a certain recommendation of either one of the approaches.

4 Conclusions and Synthesis

A framework for the resolution of multi-objective vehicle routing problems has been presented. After having previously analyzed the behavior of the approach in

interactive scenarios where the decision maker changes his/her preferences [12], we have been able to see that the clustering of customers to vehicles plays an important role in the resolution process, particularly when having to adapt to changing inputs.

The current investigation therefore compared different clustering approaches in multi-objective vehicle routing. The experimental investigation have been based on benchmark instances taken from the literature.

In conclusion, it is possible to state that the *alternative cost* assignment logic is preferable to the *minimum price* approach in most cases. However, as counterexamples can be found, we cannot entirely rule out the applicability of the otherwise weaker assignment logic. As a synthesis, both approaches could be, given appropriate time for computations, used in parallel while clearly giving priority to the alternative cost approach first.

References

1. Alba, E., Almeida, F., Blesa, M., Cotta, C., Diaz, M., Dorta, I., Gabarro, J., Leon, C., Luque, G., Petit, J., Rodriguez, C., Rojas, A., Xhafa, F.: Efficient parallel LAN/WAN algorithms for optimization. the MALLBA project. Parallel Computing 32(5–6), 215–440 (2006)
2. Beasley, J.E.: OR-library: Distributing test problems by electronic mail. Journal of the Operational Research Society 41(11), 1069–1072 (1990)
3. Cahon, S., Melab, N., Talbi, E.-G.: ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. Journal of Heuristics 10, 357–380 (2004)
4. Cahon, S., Simarik, T., Vironda, T.: A graphical user interface for multi objective optimization GUIMOO, http://guimoo.gforge.inria.fr/
5. Chitty, D.M., Hernandez, M.L.: A hybrid ant colony optimization technique for dynamic vehicle routing. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 48–59. Springer, Heidelberg (2004)
6. Cordeau, J.-F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society (52), 928–936 (2001)
7. Cordeau, J.-F., Gendrau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks (30), 105–119 (1997)
8. Di Gaspero, L., Schaerf, A.: Easylocal++: an object-oriented framework for the flexible design of local-search algorithms. Software: Practice & Experience 33(8), 733–765 (2003)
9. Ehrgott, M., Rau, A.: Bicriteria cost versus service analysis of a distribution network – a case study. Journal of Multi-Criteria Decision Analysis 8, 256–267 (1999)
10. Fink, A., Voß, S.: HotFrame: A heuristic optimization framework. In: Voß, S., Woodruff, D.L. (eds.) Optimization Software Class Libraries, ch. 4, pp. 81–154. Kluwer Academic Publishers, Boston (2002)
11. Geiger, M.J.: Genetic algorithms for multiple objective vehicle routing. In: de Sousa, P. (ed.) [24], pp. 349–353
12. Geiger, M.J., Wenger, W.: On the interactive resolution of multi-objective vehicle routing problems. In: Obayashi, et al. (eds.) [22], pp. 687–699. ISBN 3-540-70927-4

13. Geiger, M.J., Wenger, W., Habenicht, W.: Interactive utility maximization in multi-objective vehicle routing problems: A "decision maker in the loop"-approach. In: Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007), Hilton Hawaiian Village, Honolulu, Hawaii, USA, April 2007, pp. 178–184 (2007) ISBN 1-4244-0698-6
14. Gendreau, M., Bräysy, O.: Metaheuristic approaches for the vehicle routing problem with time windows: A survey. In: MIC 2003: Proceedings of the Fifth Metaheuristics International Conference, Kyoto, Japan, August 2003, pp. 1–10 (2003)
15. Hansen, P., Mladenović, N.: Variable neighborhood search. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, ch. 6. International Series in Operations Research & Management Science, vol. 57, pp. 145–184. Kluwer Academic Publishers, Boston (2003)
16. Jozefowiez, N., Semet, F., Talbi, E.-G.: Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing 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. 271–280. Springer, Heidelberg (2002)
17. Jozefowiez, N., Semet, F., Talbi, E.-G.: Enhancements of NSGA II and its application to the vehicle routing problem with route balancing. In: Talbi, E.-G., Liardet, P., Collet, P., Lutton, E., Schoenauer, M. (eds.) EA 2005. LNCS, vol. 3871, pp. 131–142. Springer, Heidelberg (2006)
18. Lee, T.-R., Ueng, J.-H.: A study of vehicle routing problems with load-balancing. International Journal of Physical Distribution & Logistics Management 29(10), 646–658 (1999)
19. Liefooghe, A., Basseur, M., Jourdan, L., Talbi, E.-G.: ParadisEO-MOEO: A framework for evolutionary multi-objective optimization. In: Obayashi,, et al. (eds.) [22], pp. 386–400. ISBN 3-540-70927-4
20. Murata, T., Itai, R.: Multi-objective vehicle routing problems using two-fold EMO algorithms to enhance solution similarity on non-dominated solutions. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 885–896. Springer, Heidelberg (2005)
21. Murata, T., Itai, R.: Local search in two-fold EMO algorithm to enhance solution similarity for multi-objective vehicle routing problems. In: Obayashi, et al. (eds.) [22], pp. 201–215. ISBN 3-540-70927-4
22. Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.): EMO 2007. LNCS, vol. 4403, pp. 3–540. Springer, Heidelberg (2007)
23. Pacheco, J., Marti, R.: Tabu search for a multi-objective routing problem. Journal of the Operational Research Society (57), 29–37 (2006)
24. de Sousa, J.P.(ed.): MIC 2001: Proceedings of the Forth Metaheuristics International Conference, Porto, Portugal (2001)
25. Potvin, J.-Y., Bengio, S.: The vehicle routing problem with time windows. Part II: Genetic search. INFORMS Journal on Computing 8(2), 165–172 (1996)
26. Potvin, J.-Y., Kervahut, T., Garcia, B.-L., Rousseau, J.-M.: The vehicle routing problem with time windows. Part I: Tabu search. INFORMS Journal on Computing 8(2), 158–164 (1996)
27. Rahoual, M., Kitoun, B., Mabed, M.-H., Bachelet, V., Benameur, F.: Multicriteria genetic algorithms for the vehicle routing problem with time windows. In: de Sousa, P. (ed.) [24], pp. 527–532

28. Savelsbergh, M.W.P.: Local search for routing problems with time windows. Annals of Operations Research 4(1), 285–305 (1985)
29. Solomon, M.M., Desrosiers, J.: Time window constrained routing and scheduling problems. Transportation Science 22(1), 1–13 (1988)
30. Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y.: A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 31(2), 170–186 (1997)

A Simple Evolutionary Algorithm with Self-adaptation for Multi-objective Nurse Scheduling

Dario Landa-silva[1] and Khoi N. Le[2]

School of Computer Science
The University of Nottingham, UK
[1]jds@cs.nott.ac.uk,
[2]kxl@cs.nott.ac.uk

Summary. We present a multi-objective approach to tackle a real-world nurse scheduling problem using an evolutionary algorithm. The aim is to generate a few good quality non-dominated schedules so that the decision-maker can select the most appropriate one. Our approach is designed around the premise of 'satisfying individual nurse preferences' which is of practical significance in our problem. We use four objectives to measure the quality of schedules in a way that is meaningful to the decision-maker. One objective represents staff satisfaction and is set as a target. The other three objectives, which are subject to optimisation, represent work regulations and workforce demand. Our algorithm incorporates a self-adaptive decoder to handle hard constraints and a re-generation strategy to encourage production of new genetic material. Our results show that our multi-objective approach produces good quality schedules that satisfy most of the nurses' preferences and comply with work regulations and workforce demand. The contribution of this paper is in presenting a multi-objective evolutionary algorithm to nurse scheduling in which increasing overall nurses' satisfaction is built into the self-adaptive solution method.

Keywords: Multi-objective, nurse scheduling, evolutionary algorithms, decoder, constraints.

1 Introduction

Producing good quality nurse schedules helps to provide better healthcare service, to improve overall job satisfaction and to make more efficient use of workforce. We are interested in tackling the nurse scheduling problem in a multi-objective fashion using an evolutionary algorithm. According to Ernst et al., the tendency in the modern workplace is to focus on individuals rather than on teams and hence, personnel schedules should cater to individual preferences [1]. This is particularly true in nurse scheduling because it is common that each nurse indicates his/her preference schedule. In our multi-objective approach, we set a target for nurse preference satisfaction and attempt to minimise the violation

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 133–155, 2008.
springerlink.com

of soft constraints related to work regulations and workforce demand. We refer to nurse scheduling as the construction of rosters for a ward of nurses over a short scheduling period (typically a few weeks). A roster can be defined as an assignment of personnel to specific shifts and/or duties. Here, a *nurse schedule* is a roster in which a line of work, made of shifts and days off, is assigned to each nurse in the ward over the scheduling period. For a discussion of other phases in the overall personnel scheduling process (e.g. demand modelling, task assignment, etc.) see [2]. Many healthcare institutions use some kind of software to aid the construction of nurse schedules but in many other cases this is still done manually [3]. For problems of considerable size, the non-automated construction of nurse schedules is time consuming, difficult and prone to errors. As Burke et al. note, "the automatic generation of high quality nurse schedules can lead to improvements in hospital resource efficiency, staff and patient safety, staff and patient satisfaction and administrative workload" [3].

Research into automated nurse scheduling has been very active in the last three decades or so. Methods applied to nurse scheduling include mathematical programming, goal programming, constraint programming, knowledge based systems, heuristics and meta-heuristics including evolutionary algorithms. Cheang et al. provide a brief literature review on the main models for nurse scheduling including types of constraints [4]. Burke et al. give a more comprehensive survey of the literature on automated nurse scheduling and classify papers with respect to nurse scheduling models and solution approaches [3]. Ernst et at. present surveys considering a wide range of personnel scheduling problems [1,2].

Nurse scheduling problems are typically over constrained and tackling them with exact optimisation methods is difficult because considering all constraints leads to complex models. Therefore, approximation algorithms have been used in many of the nurse scheduling papers in the literature. In particular, heuristics and meta-heuristics have been very popular in recent years [3]. Reasons for this are that these methods can deal with the great number of existing constraints, they can be adapted to a wide range of problems and no mathematical models are required for their implementation. Nurse scheduling is a multi-criteria problem in which typically, work regulations, workforce demand, staff preferences and efficiency of service are in some kind of conflict. Most of the research on automated nurse scheduling has been conducted in the single-objective case using an aggregating penalty function to assess the quality of schedules. Several goal programming approaches in which criteria are prioritised and targets are set for each criterion, have been reported in the literature (e.g. [5,6]). Not many applications of modern multi-objective meta-heuristics to nurse scheduling can be found in the literature (see surveys on the topic [1,3,4,7]). One of the few examples is the Pareto simulated annealing algorithm for nurse scheduling in Polish hospitals [8]. That algorithm is a population-based method in which neighbourhood exploration is carried out as in the classical simulated annealing, but the search is guided using a weighted function in order to approach the trade-off surface [9].

In this paper, we apply a multi-objective evolutionary algorithm to tackle the problem of constructing schedules for a ward of nurses in the ophthalmological unit of the QMC hospital in Nottingham, UK. Our algorithm incorporates a re-generation strategy and a self-adaptive schedule decoder. The re-generation strategy replaces dominated solutions with new offspring in order to maintain diversity and re-activate the generation of high-quality solutions when the evolutionary process stagnates. The decoder is self-adaptive because it incorporates a self-mutation operator that adapts itself to the decoding process in order to repair hard constraint violations. Section 2 describes the QMC nurse scheduling problem and outlines previous work on this problem. Section 3 gives details of the solution encoding and its relation to the nurse's preference schedule. Section 4 describes our multi-objective approach in which nurse's preferences play a central role. Section 5 gives details of the self-adaptive schedule decoder incorporated in our algorithm. Section 6 presents experiments and results while final remarks are given in Section 7.

2 The QMC Nurse Scheduling Problem

2.1 Problem Description

The problem is to construct schedules for a ward of nurses in the Queens Medical Centre (QMC) in Nottingham, UK. The scheduling period is 28 days long. A ward typically consists of 20 to 30 nurses. Cover is required on a 24 hour basis, 7 days a week. Each nurse works either on a part-time or a full-time basis. Nurses are classified in a hierarchy according to their qualifications. Some nurses receive special training according to their ward. There are three types of shift: early, late and night. The early shift is from 07:00 to 14:45 counting for seven and a half hours (7.5 hours). The late shift is from 13:00 to 21:15 counting for seven and a half hours (7.5 hours). The night shift is from 21:00 to 07:15 counting for ten hours (10 hours). Occasionally, nurses indicate in their preference schedules the starting and finishing time that they prefer to work instead of one of the above 'normal shifts'. In that case, the 'unusual shift' is considered as the 'normal shift' (early, late or night shift) that covers most of the hours of the 'unusual shift'. For example, an 'unusual shift' from 09:00 to 17:00 is considered as an early shift. If the 'unusual shift' is equally spread over two adjacent 'normal shifts', one of these 'normal shifts' is uniformly chosen at random. For example, an 'unusual shift' from 17:00 to 01:15 can be considered as a late or as a night shift. The coverage demand, i.e. the required number of nurses with specific qualifications and training, is different for each shift. Nurses specify their individual working preferences (e.g. days off, preferred shifts, etc.) for each scheduling period. A number of working regulations (including nurses' annual leave) must be satisfied. Then, the problem is to construct a schedule that meets the workforce demand, satisfies all regulations and meets as many individual preferences as possible. The QMC nurse scheduling problem includes the most common constraints in

nurse scheduling literature as identified in [4]. We formulate this problem as the ordered pair:

$$NRP = \langle Nurses, C \rangle$$

where $Nurses = \{N_i : 1 \leq i \leq n\}$ is a set of n nurses and C is a set of constraints. Constraints in C can be hard (must be satisfied) or soft (should be satisfied). A nurse N_i is defined as follows:

$N_i = \langle NurseDetail_i, NursePreference_i, NurseSchedule_i, GeneSequence_i \rangle$
$NurseDetail_i = \langle Contract_i, Qualification_i, Trained_i, Hours_i \rangle$
$Contract_i \in \{FullTime, PartTime\}$ nurse N_i is full-time or part-time.
$Qualification_i \in \{RN, EN, AN, SN\}$ nurse N_i belongs to one of four qualification categories: registered (RN), enrolled (EN), auxiliary (AN) and student (SN). RNs and ENs are classified as qualified (QN) while QNs and ANs are both employed (PN). Qualified nurses, QNs, can receive additional training specific to the ward that they work in.
$Trained_i \in \{NoTrained, Trained\}$ in the ophthalmological ward, a nurse can receive eye-training.
$Hours_i \in \mathbf{N}^+$ is the number of contracted hours for nurse N_i, for full-time nurses $Hours_i$ is 75 hours per fortnight, for part-time nurses $Hours_i$ is per week and as specified in their individual contract.
$NursePreference_i = \{p_{i,j} : 1 \leq j \leq NoOfDays\}$ is the nurse's preference schedule for the scheduling period, where $NoOfDays$ is the length of the scheduling period, 28 in the QMC problem, and $p_{i,j} \in \{AnnualLeave, Any, DayOff, Early, Late, Night\}$ is the nurse's preference for day j, Any indicates no specific preference.
$NurseSchedule_i = \{s_{i,j} : 1 \leq j \leq NoOfDays\}$ is an individual nurse's schedule, i.e. a string containing the assigned shift for each day in the scheduling period, where $s_{i,j} \in \{AnnualLeave, DayOff, Early, Late, Night\}$. A ward schedule for the QMC problem is a collection of n individual nurse schedules.
$GeneSequence_i = Permutation\{shift : 1 \leq shift \leq NoOfShifts\}$ is the gene representation used in the evolutionary algorithm implemented in this paper, where $NoOfShifts = NoOfDays * 3$ i.e. the total number of shifts in the scheduling period. This representation is illustrated in detail in Section 3.

2.2 Hard Constraints

OneShiftADay. A nurse works at most one shift (*Late*, *Early*, *Night*) per day.
MaxHours. Nurses can work a maximum number of hours (given by $Hours_i$) over a period of time according to their individual contract.
MaxDaysOn. The maximum number of consecutive days that a nurse can work, which is 6. This constraint guarantees regular breaks for nurses.
MinDaysOn. The minimum number of consecutive days that a nurse can work. This value is normally 2 for full time nurses. It is not applicable for most part time nurses because of the fewer number of shifts that they work.

Succession. Defines illegal shift combinations for nurses. A *Night* shift must not be followed by an *Early* shift.

HardRequest. Defines nurses' requests that must be satisfied. For example, *annual leave* requests in the preference schedule are considered hard requests.

2.3 Soft Constraints

SoftRequest. Defines nurses' requests that are desirable but might be violated. In the QMC problem, these requests are typically for working on specific shifts (*Early*, *Late*, *Night*, *DayOff* and 'unusual shifts').

SingleNight. A nurse should not be assigned an individual *Night* shift. Nurses at the QMC ward prefer to work night shifts in blocks of two or more. This applies to all full time nurses and certain types of part time nurses whose individual contracts are at least 20 hours a week.

WeekendSplit. Nurses prefer to work both days of the weekend or none at all.

WeekendBalance. The maximum number of weekends that nurses may work over the scheduling period. In the QMC ward, nurses may not work more than 3 out of 4 consecutive weekends.

Coverage. A certain number of nurses with specific qualifications and specific training should be assigned to particular shifts as shown in Table 1.
It should be noted that it is not necessary to assign 6 different nurses to the *Early* shift to meet the *Coverage* requirements. This demand can be satisfied with only 4 nurses if all of them are *qualified*, one of them is *registered* and one of them has received *eye-training*.

CoverageBalance. The number of nurses assigned to each shift over the scheduling period should be evenly distributed. Any surplus/shortage of nurses over the scheduling period should be kept to a minimum. This constraint prevents an excessive number of nurses being assigned to a particular shift while having a shortage of nurses in other shifts.

Table 1. *Coverage* demand of nurses in each shift

	Early	Late	Night
QNs	4	3	2
RNs	1	1	0
ETs	1	1	1

Table 2. Measurement of *CoverageBalance*

QNs	Early	Late	Night
Demand	4	3	2
Assigned	4	4	1
Difference	0	1	-1

All hard constraints must be satisfied for a schedule to be feasible. We assess the quality of a feasible schedule by measuring the violation of the six soft constraints but always taking into account the preferences expressed by each nurse. To measure the satisfaction of soft constraints (with the exception of *CoverageBalance*), we simply count the number of violations of each soft constraint type. However, the violation of a soft constraint is not penalised if the shifts assigned to the nurse's schedule comply with the nurse's preferences expressed

in $NursePreference_i$. More precisely, we measure the violation of soft constraints as follows.[1]

SoftRequest$(p_{i,j}, s_{i,j})$ if the assigned shift $s_{i,j}$ is not as the nurse's preferred shift $p_{i,j}$, a penalty of 1 is applied. No penalty is applied if a working shift $p_{i,j}$ (*Early*, *Late*, or *Night*) is requested and a *DayOff* $s_{i,j}$ is assigned.

SingleNight(N_i, D) if a *Night* shift is assigned to nurse N_i on day D, and shifts different to *Night* are assigned on adjacent days ($D-1$ and $D+1$), and the assigned shifts are not as in $NursePreference_i$, a penalty of 1 is applied.

WeekendSplit(N_i, D) if nurse N_i is assigned to work only on one of days D or $D+1$ of a weekend, and the assigned shifts are not as in $NursePreference_i$, a penalty of 1 is applied.

WeekendBalance(N_i) if nurse N_i is assigned to work at least one day in each of the four weekends in the scheduling period, and the assigned shifts are not as in $NursePreference_i$, a penalty of 1 is applied.

Coverage(*shift*) if the number of nurses with specific qualifications and training assigned to a given shift is less than the coverage demand, a penalty equal to the deficit in the number of nurses assigned is applied.

CoverageBalance we measure the satisfaction of this constraint using statistical variation on the difference between the number of qualified nurses assigned to each shift and the coverage demand for qualified nurses. For example, for a schedule of 1-day and 3 shifts (*Early*, *Late*, *Night*), the difference between coverage demand and assigned nurses is calculated as in Table 2. Then, *CoverageBalance* is measured as the variation on the *Difference* set of $3 * NoOfDays$ shifts. In this example, the penalty of the *CoverageBalance* constraint has a value of $\frac{2}{3}$ (0.667).

We split the six soft constraints into four groups and each group corresponds to an objective function. Group 1 consists of the *SoftRequest* constraint measuring the level of nurse preferences satisfaction. Group 2 consists of *SingleNight*, *WeekendBalance*, and *WeekendSplit* constraints measuring satisfaction of work regulations. Group 3 consists of the *Coverage* constraint measuring shortfalls in workforce demand. Group 4 consists of the *CoverageBalance* constraint measuring the distribution of nurses in the schedule to ensure a balanced coverage for the whole scheduling period. In the QMC problem, nurses satisfaction is at the centre of the scheduling process due to the shortfall of staff in hospitals recently. Therefore, when constructing a schedule with our multi-objective approach, we pre-set a target value for objective 1 to guarantee a minimum level of staff satisfaction. Moreover, nurses' preferences are also taken into account when dealing with the other three objectives which are subject to optimisation (this is explained in detail in the following sections). The aim is to produce a set of schedules in low computational time, all with the required level of staff satisfaction but representing a set of compromise alternatives with respect to

[1] Full description of the QMC nurse scheduling problem, examples of how the violation of soft constraints is measured and data sets are available at the following url http://www.cs.nott.ac.uk/~kxl/research/QMC/qmc.html

the other three objectives. Then, the decision-maker (typically a senior nurse) can select the most appropriate schedule for the given planning period.

2.4 Previous Work on the QMC Problem

The QMC nurse scheduling problem was also tackled by Beddoe and Petrovic using a combination of case-based reasoning and tabu search concepts [10,11]. Our results are not easily comparable to those obtained by Beddoe and Petrovic because of two reasons. One is that they tackled the QMC problem in a single-objective manner attempting to find one feasible solution, while we seek a set of alternative schedules. The other reason is that they tackled a simplified version of the problem. In their study, Beddoe and Petrovic considered constraints *OneShiftADay*, *MaxHours*, *MaxDaysOn*, *MinDaysOn Succession*, *Coverage*, *HardRequest* and *SoftRequest* only. However, we applied our approach to the data sets used by Beddoe and Petrovic and results are reported in Section 6.

3 Schedule Encoding and Construction

We represent a solution (ward schedule) to the QMC nurse scheduling problem as a set of n sub-solutions. Each sub-solution is a list of *NoOfShifts* shifts corresponding to an individual nurse's schedule. In many nurse scheduling papers, the approach is to start from an empty schedule. Instead, we take the set of preference schedules into account. Nurses indicate their preferred shifts in the preference schedule ($NursePreference_i$) while the constructed schedule ($NurseSchedule_i$) indicates the actual shifts assigned. Figure 1 illustrates a preference and a constructed schedule for one nurse on a 7-day scheduling period where E, L, N, O correspond to *Early*, *Late*, *Night*, and *DayOff* shifts respectively. An empty cell in the preference schedule indicates no preference for that day. In the constructed schedule, an empty cell represents a *DayOff*. In this example, the nurse has to work in days 1 to 5 and only one of the preferred shifts (for day 2) was satisfied. We use an indirect representation in our evolutionary algorithm in which a permutation list of integers from 1 to $3 * NoOfDays$ is decoded to create an individual nurse schedule. Figure 1 also illustrates a permutation list for a 7-day scheduling period.

Starting from the left, the decoder reads a shift x from the permutation list ($1 \leq x \leq 3 * NoOfDays$) and decodes it to the corresponding day and shift (*Early*, *Late*, *Night* correspond to 0, 1, 2 respectively) in the constructed schedule as

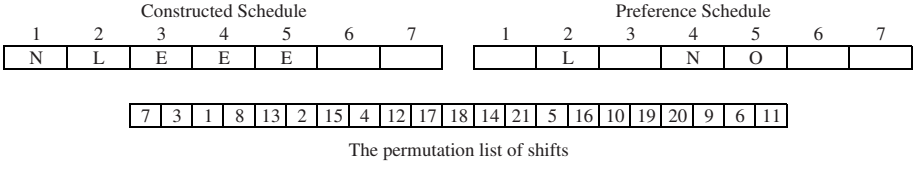


Fig. 1. Constructed Schedule, Preference Schedule and Permutation List

follows: day $D = (x-1) \operatorname{div} 3 + 1$, shift $S = (x-1) \bmod 3$. For example, $x = 7$ in the permutation list represents an *Early* shift ($S = 0$) in day 3 ($D = 3$). When assigning shifts, the decoder ensures the satisfaction of all hard constraints (see Section 5). The decoder stops assigning shifts to the nurse's schedule when the end of the permutation list is reached or the total number of working hours is within a threshold τ given by:

$$MaxHours - MinShiftLength < \tau \leq MaxHours$$

where *MaxHours* is as defined in Section 2 and *MinShiftLength* is the length of the shortest shift (7.5 hours in the QMC problem).

4 The Proposed SEAMO-R Algorithm

SEAMO, the Simple Evolutionary Algorithm for Multi-Objective Optimisation was proposed by Valenzuela who also showed that this algorithm outperforms other state-of-the-art Pareto-based evolutionary algorithms on the multiple knapsack problem [12] with respect to *the size of space covered* $\mathcal{S}$ and *the coverage of two sets* $\mathcal{C}$ indicators ($\mathcal{S}$ and $\mathcal{C}$ are defined in [13]). SEAMO uses a steady-state population and a simple elitist replacement strategy. The algorithm chooses each member of the population, in turn, to be the first parent and a second parent is chosen at random. Offspring is produced by applying cycle crossover [14] on the two parents followed by a single mutation. If the offspring's objective vector improves on any *best-so-far* objective function, it replaces one of the parents and the objective's *best-so-far* is updated. Otherwise, if the offspring dominates one of the parents, it replaces that parent (unless it is a duplicate, then the offspring is deleted). SEAMO2, an updated version of SEAMO was presented later and it was observed that a more elaborate replacement strategy improved the performance of the algorithm on both combinatorial and continuous multi-objective problems [15]. SEAMO2 is different from SEAMO in that SEAMO2 allows the offspring to replace an individual in the population that it dominates if the offspring and both parents are mutually non-dominated while in SEAMO the offspring was discarded in this case.

We implemented both versions of SEAMO on the QMC problem and observed that a major drawback was that no good offspring was generated after only few generations. Therefore, we designed a re-generation strategy that activates the production of high-quality offspring to tackle this stagnation issue. In this paper, we exploit the SEAMO's concept and propose SEAMO-R (simple evolutionary algorithm for multi-objective with re-generation), a variation of the SEAMO approach, to tackle the QMC nurse scheduling problem. Figure 2 illustrates the SEAMO-R algorithm.

The population *re-generation strategy* (Regenerate population in Figure 2) reactivates the generation of high-quality offspring when the evolutionary process stagnates. Improving the population in SEAMO-R relies entirely on the replacement strategy and the re-generation strategy. The purpose of the re-generation strategy is to maintain diversity in the population and to produce good offspring.

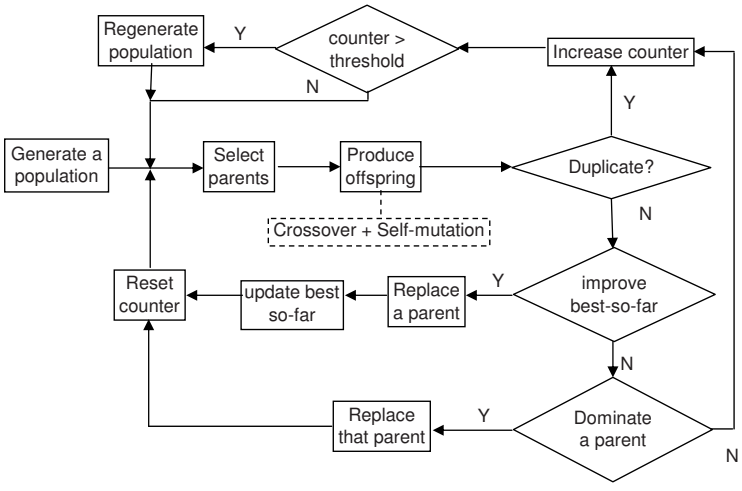


Fig. 2. The SEAMO-R algorithm

This strategy replaces a portion of dominated solutions with new solutions. The *Regeneration-Probability* parameter controls the rate of replacing a dominated solution with a new solution. A probability of 1 means that all dominated solutions will be replaced. A probability of 0 means that no dominated solutions will be replaced and the re-generation mechanism is deactivated. This replacement process is triggered if after a number of evaluations given by *Regeneration-Rate*, there is no replacement of solutions in the population with offspring. The *Regeneration-Rate* parameter is important on the performance of SEAMO-R. If the *Regeneration Rate* is too high, the population will be frequently re-generated and the population hardly evolves. If the *Regeneration-Rate* is too low (or zero as in SEAMO), the evolutionary process falls into stagnation and is very difficult to produce offspring to replace solutions in the population. The *Regeneration-Rate* is highly dependant on the *Population-Size* and the *Regeneration-Probability* (we describe later how we set these parameters in our experiments). Full details of SEAMO-R are given in the pseudocode below:

Procedure SEAMO-R
Begin
Regeneration-Rate = ⟨*pre-defined by user*⟩
Regeneration-Probability = ⟨*pre-defined by user*⟩
Population-Size = P
non-improve-counter = *0*
Soft-Request-Probability = ⟨*pre-defined by user*⟩
Generate a random population of P schedules
Evaluate the objective vector for each schedule and store it
Record the *best-so-far* for each objective function
Repeat
For each schedule in the population (1st parent)
Select a second schedule at random (2nd parent)
Apply crossover to produce offspring
Decode permutation list for each nurse's schedule in the offspring
Evaluate the objective vector for the offspring

```
        If offspring's objective vector improves on any best-so-far
          Then the offspring replaces one of the parents and best-so-far is updated
                 non-improve-counter = 0
          Else If the offspring is not duplicate and dominates 1st parent (or 2nd parent)
                 Then the offspring replace that parent
                        non-improve-counter = 0
                 Else non-improve-counter = non-improve-counter + 1
               EndIf
        EndIf
        If non-improve-counter ≥ Regeneration-Rate (**Re-generation Starts Here**)
          Then replace all dominated schedules in the current population
                        with probability of Regeneration-Probability
                        by schedules generated uniformly at random
               non-improve-counter = 0
        EndIf
      Endfor
    Until stopping condition satisfied
    Print all non-dominated solutions in the final population
End
```

5 Decoder and the Hard Constraints

5.1 Self-adaptive Schedule Decoder

The performance of SEAMO-R on the QMC nurse scheduling problem is significantly better than the performance of SEAMO and SEAMO2 because of our regeneration strategy. The performance of SEAMO-R on the QMC nurse scheduling problem is further improved by using a self-adaptive decoder to handle hard constraints. SEAMO-R also incorporates a self-mutation operator that works according to the current state of the decoding process. The decoder chooses, from either the preference schedule or the offspring's genetic material, a shift *S'* to assign to day *D*. The self-mutation operator swaps the shift *S'* with the left-most shift in the offspring's gene sequence which is associated to day *D*. If the shift *S'* chosen from the preference schedule is a *DayOff* , there is no self-mutation and the decoder moves to the next shift in the gene sequence. During the decoding process, the *Soft-Request-Probability* indicates the rate at which a shift in the preference schedule is used by the decoder rather than the current shift in the offspring's genetic material. A probability of 0 means that the decoder uses the offspring's genetic material without considering the preference schedule. A probability of 1 means that the decoder uses the preferred shift first and if this shift is not suitable, the decoder then uses the offspring's genetic material. Details of the self-adaptive decoder are shown in the following pseudocode:

```
Procedure QMC-Decoder
Begin
  Repeat
    Select a nurse schedule at random
    Assign all hard requests to that nurse's constructed schedule
    For each shift S in the permutation list of this nurse
      If the nurse schedule is fully assigned (depended on nurse's MaxHours constraint)
        Then terminate the decoding process (terminate the For loop)
      EndIf
      If day D associated with this shift S is not yet assigned
        Then
          If assign(S, D) violates Succession constraint
            Then choose shift S' from Preference Schedule
```

```
                    with probability Soft-Request-Probability or from permutation list
                    (assign(S', D) does not violates Succession)
                 If assign(S', D) does not violate MaxDaysOn, MaxHours, HardRequets
                    and the Coverage demand for shift S' of day D is not exceeded
                    Then assign S' to D, and apply the self-mutation process on S'
                 EndIf
              Else
                 If assign(S, D) violates MinDaysOn constraint
                    Then choose shift S' from Preference Schedule with
                         probability Soft-Request-Probability or from permutation list
                         to assign to day D' adjacent to day D
                         (assign(S, D) and assign(S', D') does not violates Succession)
                       If assign(S, D) and assign(S', D') do not violate
                          MaxDaysOn, MaxHours, HardRequets and Coverage demands
                          for shift S of day D and shift S' of day D' is not exceeded
                          Then assign S to D, S' to D'
                               and apply the self-mutation process on S'
                       EndIf
                    Else
                       If the Coverage demand for shift S of day D is not exceeded
                          Then assign S to D
                       EndIf
                 EndIf
           EndIf
        EndIf
     EndFor
     Evaluate objective functions
  Until all nurse schedules are decoded
End
```

The adaptive strategy in our decoder aims to guide the construction of schedules taking into account all nurses' preferences. There is no guarantee that a given permutation, once decoded, will always produce the same schedule. This is because our decoder incorporates stochastic elements that help to explore different possibilities. In general, there are two ways to satisfy hard constraints when constructing a nurse schedule. One is to only accept the assignment of shifts that maintain feasibility. The other is to repair an infeasible schedule by changing the shift assigned to day D or changing the shift assigned to an adjacent day. In this paper, we use the first approach to deal with the *OneShiftADay*, *MaxHours*, *MaxDaysOn* and *HardRequest* hard constraints. We use the second approach to deal with the *Succession* and *MinDaysOn* hard constraints. We repair a violation of the *Succession* hard constraint by changing the shift assigned to day D such that a *Night* shift is not followed by an *Early* shift. We repair a violation of the *MinDaysOn* hard constraint by assigning a working shift to an adjacent day. We take care to ensure that the *Succession* hard constraint is not violated while repairing the *MinDaysOn* hard constraint.

A number of self-adaptive approaches have been proposed in the context of evolutionary algorithms. Most approaches focus mainly on adjusting parameter settings (such as the probability of mutation and recombination) and selecting the evolutionary operators (from a range of operators available) to be applied at different times during the search. For example, Meyer-Nieberg and Beyer proposed a self-adaptive *punctuated crossover* that adapts the number and location of crossover points [16] while Sereni et al. proposed a self-adaptive recombination approach in which the crossover operator is chosen at random [17]. For reviews on adaptation and self-adaption in evolutionary algorithms see [16,18,19,20]. Note

that our self-adaptive mutation operator adjusts its characteristics throughout the search process by adapting the rate of mutation, the position of mutation points and the number of mutation points. That is, our mutation strategy 'learns' the gene sequence of individuals in order to identify good genes and appropriate positions for those good genes in the gene sequence. The mutation strategy does this by shifting genes that violate hard constraints to the end of the gene sequence, and shifting genes that provoke less violations to the beginning of the gene sequence. As the search progresses, the mutation strategy gradually shifts 'promising genes' to the beginning of the gene sequence leading to a quicker decoding process and subsequently a reduction in the number of mutation points and the mutation rate. In the rest of this Section we give more details on how the self-adaptive mutation operator works.

5.2 Handling *Succession*

We illustrate how the decoder deals with the *Succession* hard constraint. The decoder assigns shifts to the constructed schedule as in Figure 3. The decoder reads shifts in the permutation list and assigns each shift to the corresponding day unless that day is already occupied or the *Succession* hard constraint is violated. Note that in Figure 3, decoding 4 from the permutation list provokes a violation of the *Succession* constraint (a *Night* shift is followed by an *Early* shift). The decoder repairs this violation using the self-mutation process. The shift to repair this violation can be chosen from the preference schedule or from the permutation list according to the *Soft-Request-Probability*.

Figure 4 illustrates the self-mutation process to repair a violation of the *Succession* hard constraint using a shift from the preference schedule. After assigning *Early* shift to day 3, *Night* shift to day 1, and *Early* shift to day 5, the decoder skips shift 2 and shift 15 in the permutation list because day 1 and day 5 in the constructed schedule are already occupied. Shift 4 is read from the permutation list. However, assigning shift 4 (*Early* shift to day 2) violates the *Succession* constraint. A high *Soft-Request-Probability* forces the decoder to use the preference schedule and then a *Late* shift is assigned to day 2. This corresponds to assigning shift 5 instead of shift 4 from the permutation list. Therefore, the self-mutation process is applied to the current permutation list to swap shift 4 and shift 5. In this case, assigning the shift in the preference schedule does not violate the *Succession* constraint. However, if assigning the shift in the preference schedule violates the *Succession* constraint, the decoder uses the permutation list as an alternative to find a suitable shift (though the decoder was forced to consider the preference schedule first). If the decoder uses a shift *DayOff* from the preference schedule, no self-mutation is required and the decoder continues with the next shift in the permutation list.

Figure 5 illustrates the self-mutation process to repair a violation of the *Succession* hard constraint using a shift from the permutation list. The decoder attempts to assign shift 12 from the permutation list (*Night* shift to day 4). However, this violates the *Succession* constraint by creating a *Night-Early* shift combination in day 4 and day 5. A low value of *Soft-Request-Probability* forces

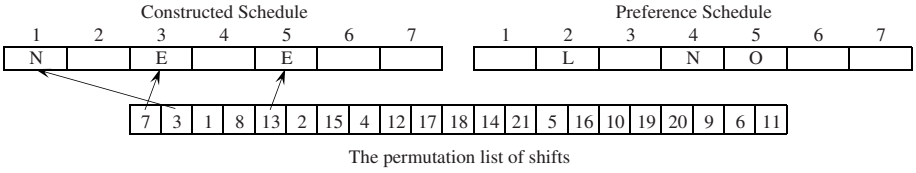


Fig. 3. The decoding process

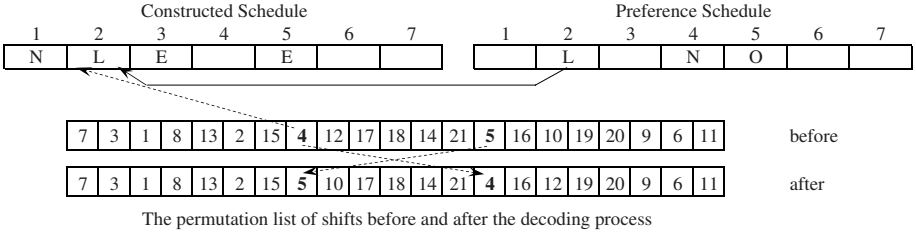


Fig. 4. Repairing the violation of the *Succession* constraint using self-mutation

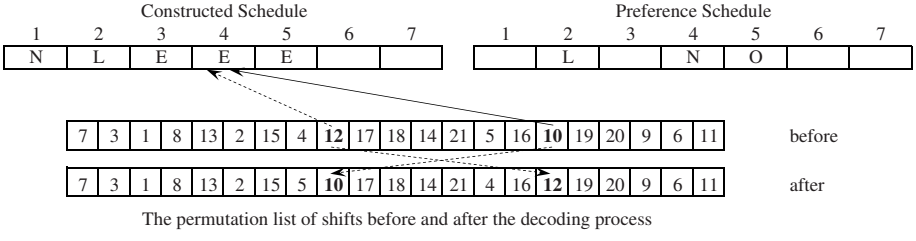


Fig. 5. Repairing the violation of the *Succession* constraint using self-mutation

the decoder to use the permutation list. The decoder searches for the first shift in the permutation list (reading from left to right) that can be assigned to day 4 without violating the *Succession* constraint. In this case, shift 10 in the permutation list (*Early* shift in day 4) is selected. The decoder assigns shift 10 and self-mutation is applied to swap shift 12 and shift 10 in the permutation list.

5.3 Handling *MinDaysOn*

To handle the *MinDaysOn* hard constraint, the decoder works on the same principle as for the *Succession* constraint. The difference is that to repair violations of the *Succession* constraint, the decoder searches for a suitable shift *S'* to assign to day *D* and the shifts assigned to adjacent days are already known. However, to repair violations of the *MinDaysOn* constraint, the decoder searches for a shift *S'* to assign to day *D'* which is adjacent to day *D* and the shift *S* assigned to day *D* is already known. First, the decoder identifies a list of suitable shifts

which do not create a shift combination that violates the *Succession* constraint. These shifts are associated with either day *D-1* or day *D+1*. From this list, the shift that appears first in the permutation list from left to right, is assigned to its associated day. The self-mutation process is then triggered. The repair of the *MinDaysOn* hard constraint is illustrated in Figure 6. The decoder continues moving to the right of the permutation list (Figure 7). Shift 12 is then assigned to the constructed schedule causing a violation of the *MinDaysOn* constraint. Therefore, the decoder has to search for an additional shift to repair the violation. The *Night* shift is assigned to day 3 in the constructed schedule (shift 9 in the permutation list). The self-mutation process swaps shift 9 with the left most shift in the permutation list associated with day 3 (shift 7 in the permutation list). In this case, we still assume that the *Soft-Request-Probability* instructs the decoder to select the additional shift from the permutation list.

Now, assume that the decoder is instructed to select an additional shift from the preference schedule when repairing violations. In Figure 8 the decoder moves right on the permutation list and assigns shift 18 (*Night* shift to day 6). An additional shift is required to repair the violation of the *MinDaysOn* constraint. The decoder searches in the preference schedule for suitable shifts. The only suitable shift in the preference schedule is the *Late* shift in day 7 (shift 20 in the permutation list) because assigning the *Early* shift to day 5 creates a violation of the *Succession* constraint (*Night-Early* shift combination in day 4 and day 5). The decoder assigns shift 20 to the constructed schedule and the self-mutation swaps shift 20 and shift 19 in the permutation list. If the decoder cannot find any suitable shift in the preference schedule to repair the violation, the decoder will look at the permutation list to find a suitable shift. For example, if the preferred shift in day 7 is the *Early* shift, there is no suitable shift in the preference schedule. Then, the decoder searches in the permutation list and finds shift 21. The self-mutation then swaps shift 21 and shift 19 in the permutation list.

Note that in the preference schedule there is a combination *DayOff-Night-DayOff* from day 2 to day 4 (Figure 9). If at any point the decoder attempts to assign *Night* shift to day 3 while day 2 and day 4 are free, then assigning *Night* shift to day 3 is not considered a violation of the *MinDaysOn* constraint. This is because the *DayOff-Night-DayOff* shift combination from day 2 to day 4 is the one indicated in the preference schedule. That is, violations of the *MinDaysOn* and *Succession* constraint are permitted if the nurse's preferences indicate so.

Besides repairing the *Succession* and *MinDaysOn* hard constraints, the decoder also attempts to minimise the number of surplus nurses in each shift. This is to reduce the number of violations of the *Coverage* constraint and equally distribute nurses amongst all shifts. This is achieved by only assigning shifts to days if the coverage demand has not been exceeded yet. Therefore, the decoder assigns shifts to days if and only if the above condition is met and the *MaxHours*, *MaxDaysOn* and *HardRequest* hard constraints are not violated. Satisfaction of *OneShiftADay* is ensured by the encoding scheme. As indicated in Section 3, the decoder assigns shifts until the end of the permutation list is reached or the total number of working hours is within the threshold τ.

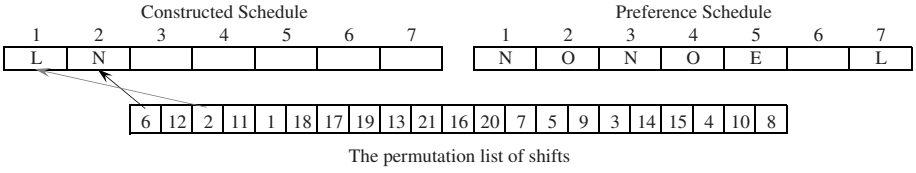


Fig. 6. Repairing the violation of the *MinDaysOn* constraint using self-mutation

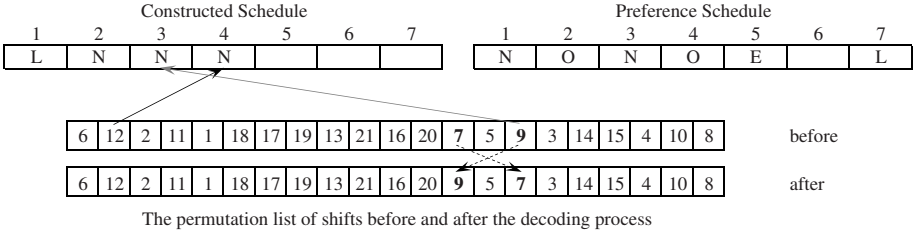


Fig. 7. Repairing the violation of the *MinDaysOn* constraint using self-mutation

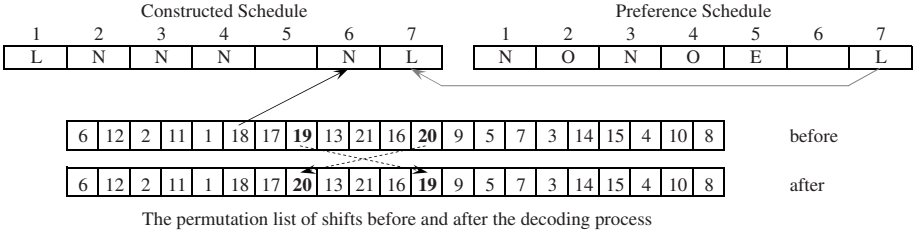


Fig. 8. Repairing the violation of the *MinDaysOn* constraint using self-mutation

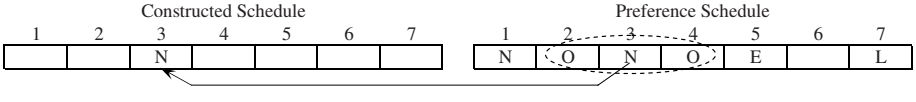


Fig. 9. Repairing the violation of the *MinDaysOn* constraint using self-mutation

6 Experiments and Results

6.1 Experimental Setting

There are 7 data sets for the QMC nurse scheduling problem, one for each scheduling period from March to September 2001. Our first experiment was to identify appropriate parameter settings for SEAMO-R. We set $RegenerationProbability = 0.75$ and $SoftRequestProbability = 0.60$ using data from March2001 as a training set. The *SoftRequestProbability* is set according to the required level of nurses'

preferences satisfaction. We performed 30 independent runs. Each run took between 4 and 5 minutes on a 2.2GHz AMD Opteron x86 64-bit Processor with Linux O/S (SuSe 9.0). We used values of *RegenerationRate* set to of 100, 200, 500 and 750. The *PopulationSize* was set to values of 100, 200 and 500 with the *NumberOfIterations* set to 15000, 7500 and 3000 generations respectively. Our preliminary results suggested that SEAMO-R performs the best when using a $PopulationSize = 200$, $NumberOfIterations = 7500$ and $RegenerationRate = 500$.

6.2 Performance of SEAMO-R

We carried out 30 runs with the above parameter settings for each of the other 6 data sets (April2001 to September2001). We found around 7 non-dominated schedules in each run. We calculate the similarity between two ward schedules as follows. Consider shift $s^1_{i,j}$ in schedule 1 and shift $s^2_{i,j}$ in schedule 2, where $1 \leq i \leq n$ (n is the number of nurses), $1 \leq j \leq NoOfDays$. Then:

$$s^1_{i,j}, s^2_{i,j} \in \{O, E, L, N\} \text{ (where } O \text{ is } DayOff)$$

$$matched_{1,2} = \left|\left\{(i,j) \mid s^1_{i,j} = s^2_{i,j} \wedge (s^1_{i,j} \neq O \vee s^1_{i,j} \neq O)\right\}\right|$$

$$unmatched_{1,2} = \left|\left\{(i,j) \mid s^1_{i,j} \neq s^2_{i,j} \wedge (s^1_{i,j} \neq O \vee s^1_{i,j} \neq O)\right\}\right|$$

$$Similarity_{1,2} = \frac{matched_{1,2}}{matched_{1,2} + unmatched_{1,2}}$$

The average similarity for the whole set of non-dominated solutions in the final population is calculated as the mean of all similarities between each pair of non-dominated solutions.

Table 3. Average results produced by SEAMO-R for the QMC problem

Period	ND	Obj2	Obj3	Obj4	Obj1	AverSimil
March2001	7.367	4.149	6.363	0.274	89.9%	78.4%
April2001	7.700	3.633	10.282	0.276	89.1%	79.3%
May2001	7.033	2.924	17.779	0.249	90.3%	85.2%
June2001	3.933	1.164	40.619	0.259	92.9%	91.5%
July2001	7.900	3.977	41.202	0.300	87.4%	80.1%
August2001	7.200	4.171	10.277	0.246	90.2%	81.8%
September2001	7.467	3.357	21.755	0.267	89.3%	84.0%

To illustrate the overall quality of the schedules generated with our approach, we show in Table 3 the 'average results' for each data set. *ND* is the average number of non-dominated solutions in the final population over the 30 independent runs. *Obj1* is the level of nurses' preferences satisfaction and is measured as a percentage as follows:

$$Obj1 = \frac{\text{Total number of requests - } SoftRequest \text{ violation}}{\text{Total number of requests}}$$

Obj2 is the total number of violations of the *SingleNight*, *WeekendSplit* and *WeekendBalance* constraints. *Obj3* is the number of violations of the *Coverage* constraint. *Obj4* is the number of violations of the *CoverageBalance* constraint. *AverSimil* is the average similarity over the 30 independent runs for each data set. Remember that *Obj1* is set as target while the other three objectives are subject to optimisation. We computed these results using the set of all non-dominated solutions obtained in the 30 runs for each data set. Details of all constructed non-dominated schedules are available from the authors. We note that given the highly constrained nature of nurse scheduling problems, it is often very difficult to find feasible schedules. This was the case in the QMC problem too and hence the need for the self-adaptive decoder and the re-generation strategy. Therefore, it was not surprising that relatively few non-dominated solutions were obtained by the end of each run. However, this number of schedules is adequate because it would be very difficult for a senior nurse to select a ward schedule from a larger set of alternatives. It is also important to note that the similarity between non-dominated schedules is high because our approach seeks to match the nurse preference schedules according to the *Soft-Request-Probability* set by the user.

Table 3 shows that the average nurses' preference satisfaction is approximately 90% for all 7 data sets. In our results, the preference satisfaction of each non-dominated solution in the final population is in the range $90\% \pm 3\%$. One can easily realise that the values for *Obj3* (violations of the *Coverage* constraint) are quite different amongst the 7 data sets. This value is quite low for March2001, April2000 and August2001 whereas for June2001 and July2001 it is noticeably high. A close examination of the data sets reveals that this is because the number of available staff-hours is quite different amongst the 7 instances. We estimate the number of available staff-hours for each of the data sets as follows: March2001 (2200), April2001 (2100), May2001 (1950), June2001 (1700), July2001 (1700), August2001 (2100) and September2001 (1930). Taking into account the coverage demand of *qualified nurses (QNs)* (see Table 1), it can be estimated that there must be at least 2030 staff-hours available to fulfill this demand. However, it is difficult to fulfill this requirement with exact 2030 staff-hours due to the existence of other constraints and individual requests. We estimate that there should be about an extra 150 staff-hours in order to minimise the number of violations. For those months with a shortage in available staff-hours (e.g. June2001 and July2001), such shortage affects mainly the provision of qualified nurses to the *Night* shift and this contributes to violations of the *Coverage* constraint. This is because with a limited number of available staff-hours, the coverage demand in shifts *Early* and *Late* is satisfied first as these shifts count for 7.5 hours. However, the *Night* shift counts for 10 hours and satisfying the coverage demand in this shift required extra staff-hours. That explains why with about 400 extra staff-hours, the number of violations of the *Coverage* constraint (*Obj3*) in March2001 is about 35 less than the number of violations in June2001 or July2001. Note that the number of violations in *Obj2* (total violations of *SingleNight*, *WeekendSplit* and *WeekendBalance*) is very low, only between 2 and 3 violations within a full schedule.

Table 4. Performance of SEAMO, SEAMO2 and SEAMO-R on the QMC problem

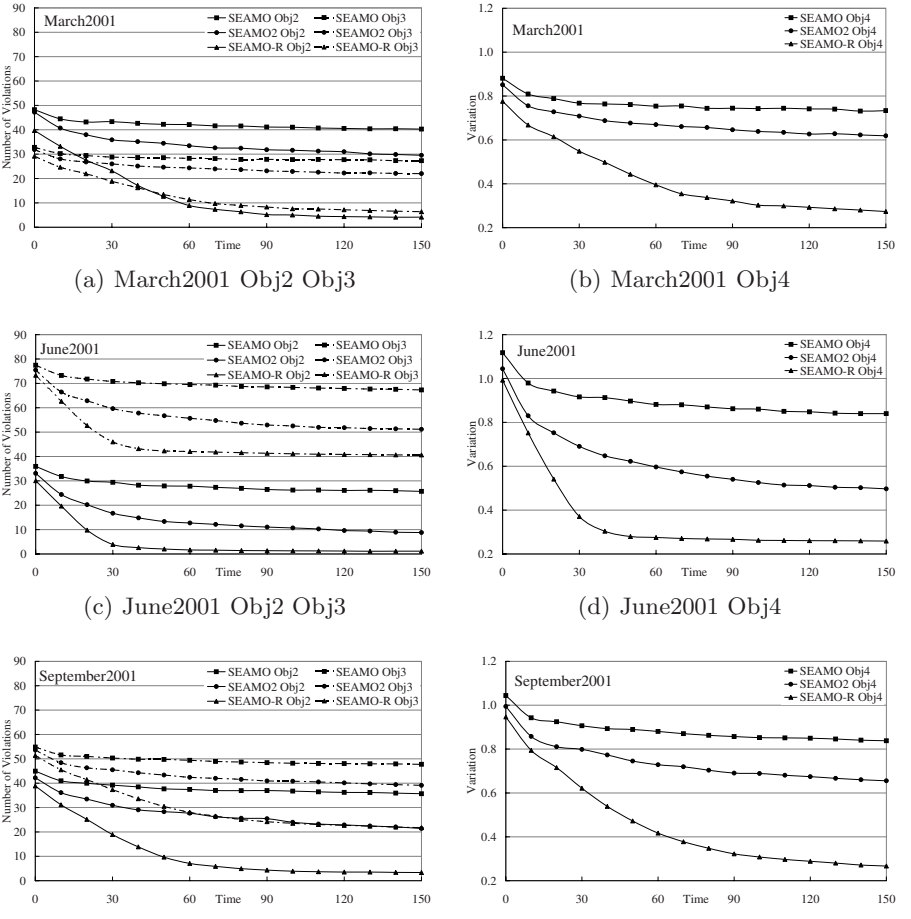
Period	SEAMO			SEAMO2			SEAMO-R		
	Obj2	Obj3	Obj4	Obj2	Obj3	Obj4	Obj2	Obj3	Obj4
Mar2001	40.274	27.320	0.734	29.522	22.001	0.619	4.149	6.363	0.274
Apr2001	40.301	33.858	0.778	29.045	28.018	0.666	3.633	10.282	0.276
May2001	32.471	44.083	0.831	22.076	36.695	0.672	2.924	17.779	0.249
Jun2001	25.691	67.312	0.840	8.791	51.145	0.497	1.164	40.619	0.259
Jul2001	27.824	63.776	0.844	15.963	55.121	0.637	3.977	41.202	0.300
Aug2001	38.590	34.849	0.779	26.796	28.996	0.658	4.171	10.277	0.246
Sep2001	35.686	47.760	0.838	21.423	39.126	0.656	3.357	21.755	0.267

6.3 Comparison with SEAMO and SEAMO2

Our next set of experiments was to compare SEAMO-R against SEAMO and SEAMO2 on the QMC problem in order to assess the contribution of our re-generation strategy and self-adaptive decoder on the good results obtained. We incorporated the self-adaptive decoder into SEAMO and SEAMO2 for these experiments. Average results are presented in Table 4 and it is clear that SEAMO-R outperforms SEAMO and SEAMO2.

We also examined the performance of SEAMO-R, SEAMO and SEAMO2 throughout the search process. We traced the evolution of the average values for the set of non-dominated solutions at every 50 generations in each of the 30 runs and for all 7 data sets. In Figure 10, we only present graphs for 3 data sets, March2001, June2001, September2001 which are representative of all our results. The graphs show that SEAMO-R quickly outperforms SEAMO and SEAMO2 and overall, SEAMO-R improves the quality of the non-dominated solutions very rapidly. While the replacement strategy in SEAMO2 helped this algorithm to outperform SEAMO on multi-objective benchmark problems [15], our experiments show that our re-generation strategy contributes substantially to the good results obtained on the highly constrained QMC nurse scheduling problem.

Regarding multi-objective optimisation, we evaluate the Pareto fronts produced by SEAMO, SEAMO2, SEAMO-R using two metrics, *size of the space covered* $\mathcal{S}$ and *coverage of two sets* $\mathcal{C}$, proposed in [13]. The $\mathcal{S}$ hypervolume metric is scaled as the percentage of the volume created by the origin and the reference point (100, 100, 3) with respect to (*Obj2*, *Obj3*, *Obj4*). The reference point is estimated using the average objective vector's value of the non-dominated solutions in the initial population. With respect to the *coverage* metric $\mathcal{C}$, all non-dominated solutions in the final population of SEAMO-R dominate the ones of SEAMO and SEAMO2 for all 7 data sets. Figure 11 measure the percentage of non-dominated objective space. The horizontal axes present SEAMO, SEAMO2, SEAMO-R. The *size of the space covered* produced by SEAMO-R is much better than the one of SEAMO and SEAMO2 for all 7 data sets. As it can be seen, the results for June2001 and July2001 are not as good as March2001 because of the shortage of available staff-hours which was explained above.



(a) March2001 Obj2 Obj3 (b) March2001 Obj4

(c) June2001 Obj2 Obj3 (d) June2001 Obj4

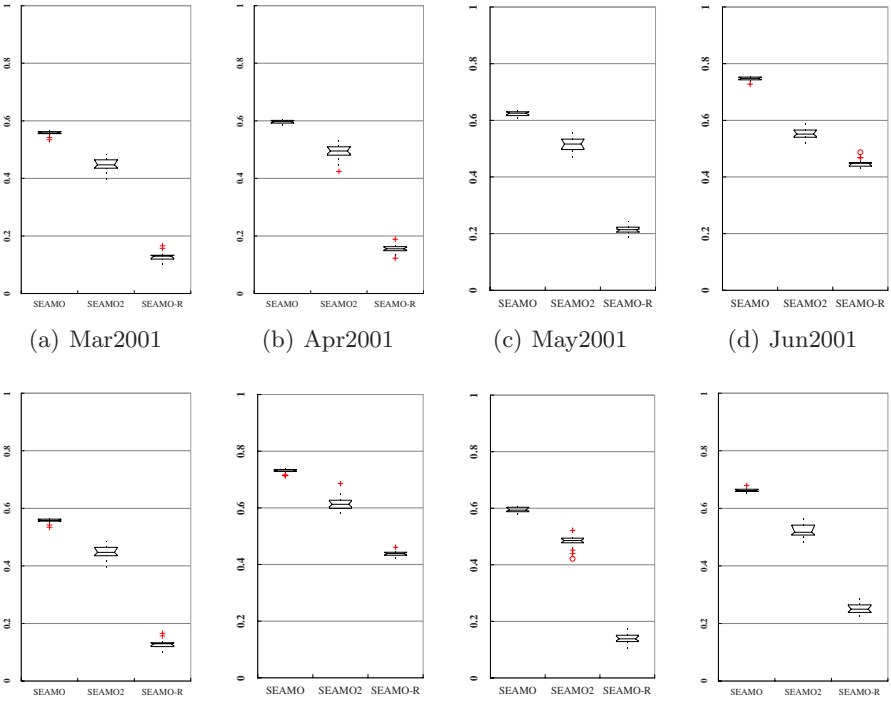
(e) September2001 Obj2 Obj3 (f) September2001 Obj4

Fig. 10. Performance of SEAMO-R, SEAMO and SEAMO2 on the QMC problem

6.4 Selecting a Ward Schedule

Although we recorded all non-dominated solutions in the final population for each run, here we simulate the decision-making process of choosing one schedule from the set of alternatives. We assume that a 'best schedule' is chosen based on the following priority:

1. number of violations of *SingleNight*, *WeekendSplit*, *WeekendBalance* (*Obj2*)
2. number of violations of *Coverage* (*Obj3*)
3. the penalty for *CoverageBalance* (*Obj4*)
4. the overall nurses' preferences satisfaction *SoftRequest* (*Obj1*)



(a) Mar2001 (b) Apr2001 (c) May2001 (d) Jun2001

(e) Mar2001 (f) Jul2001 (g) Aug2001 (h) Sep2001

Fig. 11. Performance of SEAMO-R, SEAMO and SEAMO2 on the QMC problem based on *size of the uncovered space* $\mathcal{S}$

However, different decision makers could use different priorities and a different schedule from the obtained non-dominated set would be chosen. We present the objective values for these 'best schedules' in Table 5.

Table 5. A selected 'best schedule' for each data set

Period	Obj2	Obj3	Obj4	Obj1
March2001	2.567	5.8	0.271	89.7%
April2001	2.000	9.767	0.275	88.9%
May2001	2.267	16.5	0.231	90.1%
June2001	0.867	39.977	0.261	92.8%
July2001	1.900	39.900	0.318	87.2%
August2001	3.467	8.233	0.217	90.0%
September2001	2.567	20.167	0.253	89.2%

Table 6. Average results of SEAMO-R on Beddoe and Petrovic data sets [10,11]

	SEAMO-R		CABAROST (CB-OBJ-TL-R10)	
Period	Obj3	Obj1	Obj3	Obj1
March2001	2.600	88.6%	0.100	90.1%
April2001	4.900	89.3%	0.000	90.7%

6.5 Previous Results on the QMC Problem

As it was mentioned in Subsection 2.4, Beddoe and Petrovic used a simplified version of the QMC problem and tackled it in a single-objective manner [10,11]. We applied our SEAMO-R approach to the data sets (March2001 and April2001) used by Beddoe and Petrovic and results are reported in Table 6. We can see that the results obtained by SEAMO-R are slightly worse than those reported by Beddoe and Petrovic. However, note that the number of violations of the *Coverage* constraint (*Obj3*) produced by SEAMO-R on the Beddoe and Petrovic data sets is only slightly better than the number of violations on the data sets of this paper (Table 3), although the later data sets correspond to much more constrained instances. Full details of the comparison with the work of Beddoe and Petrovic are available in [21]. In order to facilitate further research and comparison with our results, we make the QMC problem instances available in the web page mentioned in Section 2.

7 Final Remarks

We have presented a multi-objective evolutionary approach to tackle a real-world nurse scheduling problem in which the satisfaction of staff preferences drives the search for non-dominated solutions. We described an adaptation of the Simple Evolutionary Algorithm for Multi-objective Optimisation (SEAMO) to tackle a nurse scheduling problem with four objectives. One of the objectives is set as a target while the other three are subject to optimisation. In our multi-objective approach, we have grouped soft constraints in a manner that is meaningful to the decision-maker (usually a senior nurse). The target objective is associated to the satisfaction of nurses' preferences. The other three objectives are associated to 1) meeting work regulations, 2) meeting coverage demand and, 3) ensuring balanced coverage demand for the whole scheduling period. We developed a re-generation strategy to aid diversification and a self-adaptive decoder to repair constraint violations. These two mechanisms are driven by the target level of nurses' preferences satisfaction which can be set by the decision-maker. The resulting algorithm is SEAMO-R, a Simple Evolutionary Algorithm with Re-generation for Multi-objective Optimisation. The re-generation strategy replaces dominated solutions with new ones to avoid stagnation. The self-adaptive decoder uses the nurse preference schedule, a random permutation of shifts and a self-mutation operator to construct schedules and maintain feasibility. Our

results show that SEAMO-R produces sets of good quality of feasible and non-dominated ward schedules for the QMC nurse scheduling problem.

References

1. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D.: An annotated bibliography of personnel scheduling and rostering. Annals of Operations Research 127, 21–144 (2004)
2. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: a review of applications, methods and models. European Journal of Operational Research 153, 3–27 (2004)
3. Burke, E.K., De Causmaecker, P., Vanden Berghe, G.: The state of the art of nurse scheduling. Journal of Scheduling 7, 441–499 (2004)
4. Cheang, B., Li, H., Lim, A., Rodrigues, B.: Nurse rostering problems: a bibliographic survey. European Journal of Operational Research 151, 447–460 (2003)
5. Authur, J.F., Ravindran, A.: A multiple objective nurse scheduling model. AIIE Transactions 13(1), 55–60 (1981)
6. Berrada, I., Ferland, J.A., Michelon, P.: A multi-objective approach to nurse scheduling with both hard and soft constraints. Socio-Economic Planning Sciences 30(3), 183–193 (1996)
7. Landa Silva, J.D., Burke, E.K., Petrovic, S.: An introduction to multiobjective metaheuristics for scheduling and timetabling. In: Gandibleux, X., Sevaux, M., Sorensen, K., T'kindt, V. (eds.) Metaheuristic for multiobjective optimisation. Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 91–129 (2004)
8. Jaszkiewicz, A.: A metaheuristic approach to multiple objective nurse scheduling. Foundations of Computing and Decision Sciences 22(3), 169–183 (1997)
9. Czyzak, P., Jaszkiewicz, A.: Pareto simulated annealing - a metaheuristic for multiple-objective combinatorial optimization. Journal of Multicriteria Decision Analysis 7(1), 34–47 (1998)
10. Beddoe, G.R., Petrovic, S.: Combining case-based reasoning with tabu search for personnel rostering problems. Computer Science Technical Report No. NOTTCS-TR-2004-5, The University of Nottingham (2004)
11. Beddoe, G.R., Petrovic, S.: Enhancing case-based reasoning for personnel rostering with selected tabu search concepts. The Journal of The Operational Research Society (to appear, 2007)
12. Valenzuela, C.L.: A simple evolutionary algorithm for multi-objective optimization (seamo). In: IEEE World Congress on Computational Intelligence (WCCI 2002): Congress on Evolutionary Computation (CEC 2002), pp. 717–722. IEEE press, Los Alamitos (2002)
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. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the traveling salesman problem. In: Genetic Algorithms and their Application: Proceedings of the Second International Conference on Genetic Algorithms, pp. 224–230 (1987)
15. Mumford, C.L.: Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1389–1400. Springer, Heidelberg (2004)

16. 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, vol. 54, pp. 47–76. Springer, Heidelberg (2007)
17. Sareni, B., Regnier, J., Roboam, X.: Recombination and self-adaptation in multi-objective genetic algorithms. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 115–126. Springer, Heidelberg (2004)
18. Hinterding, R., Michalewicz, Z., Eiben, A.E.: Adaptation in evolutionary computation: a survey. In: 1997 IEEE International Conference on Evolutionary Computation, pp. 65–69 (1997)
19. Bäck, T.: Self-adaptation in genetic algorithms. In: Varela, F.J., Bourgine, P. (eds.) Proceedings of the 1st European Conference on Artificial Life (ECAL 1992), pp. 227–235. MIT Press, Cambridge (1992)
20. Angeline, P.J.: Adaptive and aelf-adaptive evolutionary computations. In: Palaniswami, M., Attikiouzel, Y. (eds.) Computational Intelligence: A Dynamic Systems Perspective, pp. 152–163. IEEE Press, Los Alamitos (1995)
21. Le, K.N.: An evolutionary algorithm for multi-objective nurse scheduling. Master Thesis, School of Computer Science and IT, The University of Nottingham (2006)

Individual Evolution as an Adaptive Strategy for Photogrammetric Network Design

Gustavo Olague[1], Enrique Dunn[1], and Evelyne Lutton[2]

[1] CICESE, Km. 107 carretera Tij-Eda 22860, Ensenada México
{olague,edunn}@cicese.mx
[2] INRIA Rocquencourt, Le Chesnay Cedex France
evelyne.lutton@inria.fr

Summary. This chapter introduces individual evolution as a strategy for problem solving. This strategy proposes to partition the original problem into a set of homogeneous elements, whose individual contribution to the problem solution can be evaluated separately. A population comprised of these homogeneous elements is evolved with the goal of creating a single solution by a process of aggregation. The goal of individual evolution is to locally build better individuals that jointly form better global solutions. The implementation of the proposed approach requires addressing aspects such as problem decomposition and representation, local and global fitness integration, as well as diversity preservation mechanisms. The benefit of applying the individual evolution approach for problem solving is a substantial reduction in computational effort expended in the evolutionary optimization process. This chapter shows an example from vision metrology where experimental results coincide with previous state of the art photogrammetric network design methodologies, while incurring in only a fraction of the computational cost.

Keywords: Individual Evolution, Coevolution, Photogrammetric Network Design.

1 Introduction

Individual evolution is a methodology that simplifies the complexity of an evolutionary algorithm based on the partition of the solution in smaller elements. This strategy could be seen from the standpoint of coevolution where several solutions evolve in the form of interacting coadapted subcomponents. To successfully solve increasingly complex problems, we must develop effective techniques for evolving cooperative solutions in the form of interacting coadapted subcomponents. Cooperative coevolution is implemented from the standpoint of individual evolution, which is also known as Parisian evolution. In the Parisian approach each individual should find only a part of the problem solution; then, a process of aggregation should group a suitable solution. In general, this methodology follows a strategy where explicit notions of modularity are applied to provide reasonable opportunities for solutions to evolve in the form of interacting coadapted subcomponents.

In the cooperative coevolutionary framework [2, 3] two main aspects are reported by which traditional computational intelligence approaches are not

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 157–176, 2008.
springerlink.com

entirely adequate for solving complex problems with high interactions between population members. Firstly, classical evolutionary approaches have a strong tendency to converge into a single solution in response to an increasing number of trials being allocated to observed regions of the solution space with above average fitness. As a result, computational effort is wasted in the search of a single solution. Secondly, individuals encoded by traditional computational intelligence approaches typically represent complete solutions that are evaluated in isolation. In this way, the interactions between population members are not modelled; and as a result, there is no evolutionary pressure for coadaptation to occur.

This work presents a novel photogrammetric network design strategy based on the individual evolution paradigm in which reasonable subcomponents emerge rather than being hand designed [4]. Parisian evolution provides the key concepts to allow an adequate framework to identify and represent such subcomponents, provide an environment in which they can interact and coadapt, identify local and global fitness evaluations, and create mechanisms for population diversity preservation, in which the evolutionary algorithm could be applied to solve the difficult problem of photogrammetric network design.

1.1 Previous Work

Photogrammetric network design addresses the process of placing cameras in order to perform photogrammetric tasks. There is a new interest on this topic from several communities [1, 5, 8, 11, 20, 21]. The choice of an adequate imaging geometry plays a major role in this process [7, 8]. The process, by which the best possible configuration can be automatically determined, is still an open research area [19]. This design problem offers an intricate combination of interactions between the sensor physical constraints, the mathematical modelling of the problem, as well as the numerical methods used to solve it [17]. Moreover, the lack of a widespread utilization of network design inside the photogrammetric community can be attributed to the inherent design complexity and its expensive computational requirements. Photogrammetric network design requires complex spatial reasoning about the geometrical characteristics of an object and the mathematical modelling of optical triangulation. This reasoning is by no means trivial and has been the topic of very diverse research. For example, the work of Mason [12] solves the problem of camera placement by incorporating considerable a priori knowledge into an expert system. In this way, a set of heuristics based on the theory of generic networks is used to model the decision making process. On the other hand, the work of Olague and Mohr [16] uses an evolutionary computation approach, developing a linear criterion based on the process of error propagation, which was further extended considering a bundle adjustment [15]. In this way, the required spatial reasoning is carried out by an adaptive system based on stochastic meta-heuristics that yield human competitive results. Recently, the work of Saadatseresht et al. [18] addresses the problem of improving an existing camera network by positioning additional sensing stations based on what they term “visibility uncertainty prediction modelling”. All the above works give

special attention to the usefulness of rigorous approaches such as bundle adjustment in order to characterize the quality of the photogrammetric network. In particular, the works of Fraser [6] and Olague and Mohr [16] have provided insight into how a mathematical modelling could be derived in order to simplify the network design. The aim of this work is to present a new network design simplification based on the Parisian approach.

The Parisian approach of individual evolution considers that a single individual represents a partial solution to the considered problem. Hence, a process of aggregation of multiple individuals is needed in order to arrive at a solution. We incorporate such approach to the camera network design problem by evolving a population of camera subnetworks. As a result, computational requirements are greatly reduced for individual fitness evaluation due to the reduced size of the total mathematical model. Parisian evolution is a complex optimization technique because multiple new aspects need to be considered in the evolutionary computation framework, such as: problem partitioning and representation, local fitness evaluation, global fitness evaluation and redistribution, population diversity preservation, and finally aggregation of individuals. In this paper, we attempt to combine widely accepted principals and techniques used in EC research. On the other hand, in the case of radical new concepts we provide a first solution based on the characteristics of the studied problem.

This paper is organized as follows. First, the paradigm of Parisian evolution is presented. Then, a solution to the photogrammetric network design in terms of Parisian evolution is described together with implementation details about the problem partitioning, solution aggregation, individual fitness evaluation and diversity preservation techniques. Finally, experimental results are presented followed by discussion and conclusions.

2 Parisian Evolution Paradigm

Parisian evolution originally proposed by Collet et al. [2], differs from typical approaches of evolutionary computation in the idea that a single individual of the population represents only a part of the solution. It is similar to the Michigan approach developed for Classifier Systems [10], where a solution is a rule base obtained from an evolved population of individual rule subsets. In this paradigm an aggregation of multiple individuals should be considered in order to obtain a solution to the problem being studied. This aggregation could be explicit or implicit. The motivation of such approach is to make an efficient use of the genetic search process. This is achieved from two complementary standpoints. Firstly, the algorithm discards less computational effort at the end of execution, while considering more than a single best individual as output. Secondly, the computational expense of the fitness function evaluation is considerably reduced for a single individual. The Parisian approach could be stated as cooperative coevolution with the aim to be applied in the general context of computational intelligence. The major difference with traditional cooperative coevolution is on the way of organizing the individuals. In traditional cooperative coevolution the

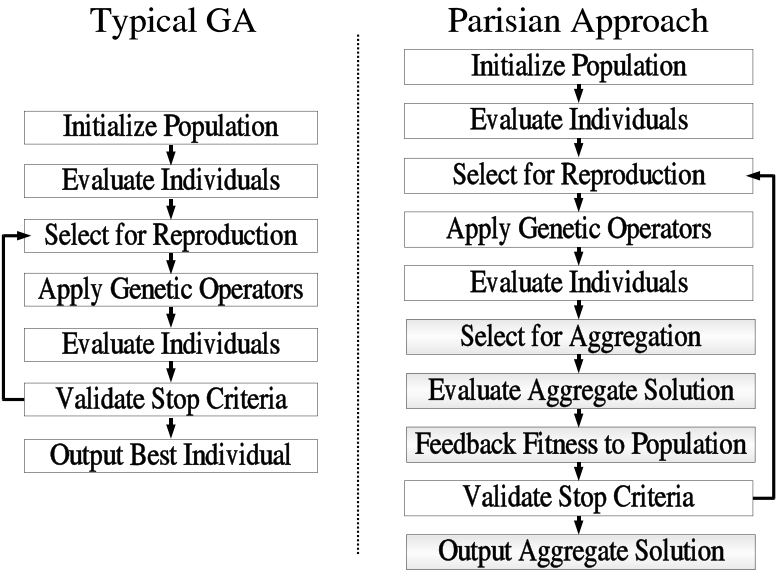


Fig. 1. Outline of our implementation of the Parisian approach. Fitness evaluation is modified in order to consider the local and global contribution of an individual.

individuals are divided in species that are genetically isolated. In other words, individuals only mate with other members of their species. Mating restrictions are enforced simply by evolving the species in separate populations. The only feedback is through a share domain model which produces a cooperative relationship. Contrary to this way of setting the framework for cooperative coevolution; the Parisian approach uses the idea of individual evolution to promote the exchange of genetic material based on the local and global fitness evaluations. This allows the coevolution of complex behaviors. Under the Parisian approach, many of the canonical aspects of evolutionary algorithms are retained, providing great flexibility in its deployment. From an algorithmic viewpoint, see Fig. 1, Parisian evolution needs four aspects in their design which are implemented with different meta-heuristics. The reader should be aware that as other meta-heuristic approaches there are not mathematical models, which could yield the optimal parameter setting in each situation. Therefore, we decide to obtain the best set of parameters of our algorithm through statistical experimentation. Thus, Parisian evolution should consider the following aspects:

1. Partial Encoding. This is the fundamental concept that is need in cooperative coevolution. The genetic representation is achieved through a number of single individuals that encode a partial solution. Therefore an individual aggregation step is necessary in order to create a complete problem solution. This process of aggregation could be explicit or implicit according to the problem being studied. This concept provides the strength to decompose

the problem by determining an appropriate number of subcomponents and the role that each subcomponent will play. In general the aggregation step has been defined by the human designer due to the difficulty of providing mathematical solutions. In the photogrammetric network design each camera network represents a subnetwork. Obviously, a single subnetwork is not enough to measure a complex object.

2. The Environment. The design of the system should provide an environment where different partial solutions interact and coadapt in order to allow the emergence of better aggregate solutions. Obviously, such a design interdicts the evolution of subcomponents without interdependencies, in order to avoid the evolution of isolate subcomponents. In the photogrammetric network design, the complex object being measured constraints the landscape in which complex interdependencies and interactions emerge.
3. Local and Global Fitness. A meaningful merit function must be designed for each partial solution. In this way, the worthiness of a single individual can be evaluated in order to estimate the potential contribution to an aggregate solution. The evolutionary engine requires a scheme for combining local and global fitness values. This could be explicit or implicit. In the photogrammetric network design the worthiness of the final configuration is a product of the interactions between the local and global evaluations carried out by an analysis of error propagation over the triangulation stage.
4. Population Diversity Preservation. In contrast to traditional computational intelligence approaches where diversity needs to be preserved only during enough time to perform a reasonable exploration of the search space; a cooperative coevolutionary approach requires that all subcomponents should be present in the final solution in order to maintain a set of complementary partial solutions.. In this respect, diversity preservation techniques need to be implemented. In evolutionary algorithms three different techniques could be applied: 1) heuristic modification of genetic operators in order to promote diversity, 2) fitness function penalization for crowded individuals, and 3) incorporation of some higher level algorithmic structure to generate and manage sub-populations. In this work, we apply the fitness sharing scheme [9].

In general terms, the Parisian approach makes the following assumptions:

1. A complete problem solution $X \in S$ can be decomposed into n components $x_i \in S'$. Moreover, there exists a mapping $T : S' \times \cdots \times S' \rightarrow S$.
2. There exists a meaningful fitness function $f_{local} : S' \rightarrow \mathbb{R}$ for evaluating each decomposed element.
3. There exists a meaningful fitness function $f_{global} : S \rightarrow \mathbb{R}$ for evaluating an aggregate solution.
4. The fitness landscape defined by f_{local} and f_{global} has sufficient structure to guide the evolutionary search process.

Under these assumptions, the evolutionary search is carried out over S' optimizing f_{local}. However, the fitness values of the evolved individuals are systematically modified in order to promote the emergence of improved composite

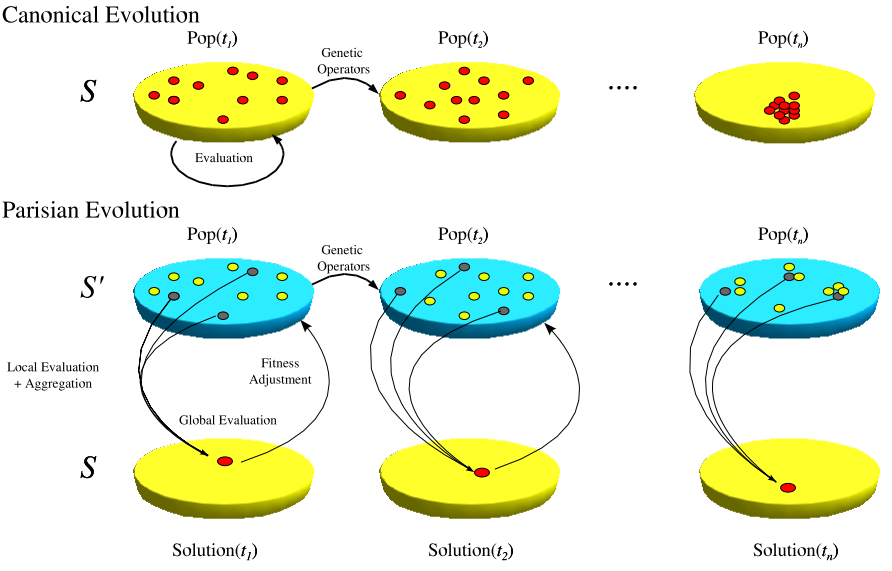


Fig. 2. Conceptual description of our Parisian approach. Interaction between different search spaces is based on adjusting the population fitness values in accordance to a global fitness evaluation.

solutions in S. This is achieved by: (1) periodically sampling the evolving population to form aggregate solutions, (2) evaluating aggregate solutions through f_{global} and (3) adjusting the fitness values of the evolving individuals in S'. Fig. 2 illustrates this process, as well as the relationship between different search spaces involved in the Parisian evolutionary approach. In this way, complex population dynamics could emerge based on the interactions between f_{local} and f_{global}.

3 Individual Evolution for Camera Planning

Camera placement can be viewed as a geometric design problem where the control variables are the spatial positioning and orientation parameters of a finite set of cameras. In order to state such design problem in optimization terms a non-linear criterion is adopted based on the process of error propagation. However, due to the imaging geometry described by the mathematical modelling of the problem a strongly constrained optimization problem emerges. In this section we will discuss the different implementation issues of the Parisian approach within the camera placement problem. The reader with interest in understanding the photogrammetric problem is advised to read the given references where complete explanations are provided.

3.1 Problem Partitioning and Representation

A viewing sphere model for camera placement is adopted in order to reduce the dimensionality of our search space. Therefore, given a fixed radius, each camera position is defined by its polar coordinates $[a_i, b_i]$. A network of M cameras is represented by a real valued vector

$$\Psi \in \mathbb{R}^{2M} \quad where \quad \alpha_i = \Psi_{2i-1}, \beta_i = \Psi_{2i} \quad for \quad i = 1, \ldots, M. \tag{1}$$

Our design problem allows the decomposition into individual elements since the complete camera network is formed by a set of homogeneous components. Nevertheless, a decision on the level of granularity of our decomposition is crucial. Here we have the choice of an individual representing a single camera or a camera subnetwork (i.e., a set of cameras). We have decided for the latter option since such an individual can be meaningfully evaluated in terms of its imaging geometry contribution to 3D reconstruction. Hence, each individual in the population represents a fixed size subnetwork of N cameras, denoted by a vector of the form

$$\psi^j \in \mathbb{R}^{2N} \quad where \quad \alpha_i = \psi^j_{2i-1}, \beta_i = \psi^j_{2i} \quad for \quad i = 1, \ldots, N; \tag{2}$$

where j is defined as the subnetwork population index. Accordingly, a complete camera network specification is given by the aggregation of J subnetworks

$$\Psi \in \mathbb{R}^{2M} = \bigcup_{j=1}^{J} \psi^j, \quad where \quad M = J \times N. \tag{3}$$

3.2 Evaluating a Camera Network Configuration

Currently, rigorous bundle adjustment procedures can be performed on a standard laptop [13]. However, the photogrammetric network design, proposed in our previous work, requires considerable computational resources due to the evolutionary computation technique [15]. Fraser [6] has shown that a simplified model called Limiting Error Propagation closely approximates the covariance matrix generated for the object point XYZ coordinates of the rigorous approach known as Total Error Propagation. This simplification is valid for strong convergent multi-station photogrammetric networks in the case of the First Order Design. The vision metrology algorithms rely on applying a set of non-linear transformations to the image measurements to compute world measurements. Since the input quantities and the transformations are uncertain, the output measurements are also uncertain. In order to determine how the uncertainty propagates from input to output of the computation chain a much faster analytical approach can be applied to estimate the 3D measurement accuracy [16]. The analytical method takes account of the fact that the 3D measurement point $\mathbf{P_j}$ is related to the input data $\mathbf{p_{ij}}$ by an analytical function f (non-linear). This relationship is approximated with a linear one by means of a first order Taylor series expansion.

By assuming noise only on the input data $\mathbf{p_{ij}}$ and not on the transformation we obtain the following relationship:

$$f(\mathbf{p}) = f(E[\mathbf{p}]) + \frac{\partial f(E[\mathbf{p}])}{\partial \mathbf{p}}(\mathbf{p} - E[\mathbf{p}]) + \Theta(\mathbf{p}) \ , \tag{4}$$

from which, ignoring the second order terms, it is easy to compute the mean value of the output measurements and consequently the covariance of the measurements:

$$\Lambda\mathbf{P} \simeq \frac{\partial f(E[\mathbf{p}])}{\partial \mathbf{p}} \ \Lambda\mathbf{p} \ \frac{\partial f(E[\mathbf{p}])}{\partial \mathbf{p}}^T \ . \tag{5}$$

Local Fitness Evaluation

This criterion estimates the uncertainty of the measurements that will be obtained by the camera network. In general such approach is applied over a complex object considering all cameras concurrently. Since in our representation we are working with camera subnetworks, it is unlikely that any single individual successfully captures the completed 3D object denoted by the whole set of 3D points $\mathbf{Q}$. Hence, the object is also partitioned into R disjoint regions or subsets of points, in such a way that $\mathbf{Q} = \bigcup_{i=1}^{R} \mathbf{P}_i$. A single region of the object is considered visible by a camera network if at least two cameras triangulate it (i.e., there are no occlusions) and for each of these cameras the incidence angle constraint for 3D reconstruction is satisfied. Hence, the visibility of a camera subnetwork ψ^j is limited to a subset of the whole object, expressed by $\mathbf{V}(\psi^j) \subseteq \mathbf{Q}$. These values are calculated a priori and stored in a database for on-line query during the optimization procedure. Accordingly, we define the visibility constraint as follows:

$$C_{vis}(\psi^j, \mathbf{P}_i) = \begin{cases} 1 & \text{si } \mathbf{P}_i \subset \mathbf{V}(\psi^j) \\ 0 & \text{otherwise .} \end{cases}$$

The reasoning of such local fitness formulae is that subnetworks which provide greater object coverage with higher precision should have higher fitness values. Hence, the uncertainty for each of the sets $\mathbf{P}_i$ is evaluated for a single individual ψ^j accordingly to Eq. 5, discarding the portions of the object not sensed by such a subnetwork. In order to promote camera subnetworks that provide greater coverage, the local fitness value is proportional to the number of regions $\mathbf{P}_i$ sensed by such subnetwork, which is expressed as $\#[\mathbf{V}(\psi^j)]$. Thus, we define the local fitness as follows

$$f_{local}(\psi^j) = w s_1 \frac{\#[\mathbf{V}(\psi^j)]}{\max f_1(\psi^j, \mathbf{P}_i)} + \frac{(1-w)\, s_2}{f_2(\psi^j)} \quad \forall \mathbf{P}_i : C_{vis}(\psi^j, \mathbf{P}_i) = 1, \tag{6}$$

where s_1 and s_2 are scale factors applied to each criterion, while the parameter w controls the weight assigned to each criterion. In this formulae, $f_1(\psi^j, \mathbf{P}_i)$ represents the 3D reconstruction uncertainty of a given region $\mathbf{P}_i$, under observation by a subnetwork parameterized by ψ^j. Note that $f_1(\psi^j, \mathbf{P}_i) > 0$, $\forall \mathbf{P}_i : C_{vis}(\psi^j, \mathbf{P}_i) = 1$. In the cases where an individual network does not cover any object region, its fitness is simply set to $f_{local}(\psi^j) = 0$.

Global Fitness Evaluation

Once the local fitness of each individual has been evaluated, a process of aggregation is needed to obtain a solution for the camera network design problem. In order to achieve it, a selection of a group of individuals from the population must be realized. Accordingly, at each generation t an aggregate solution $\Psi(t)$ is obtained for global fitness evaluation. This global evaluation uses the same criterion of local fitness evaluation, which also combines f_1 and f_2 through the following expression:

$$f_{global}(\Psi(t)) = \frac{w\, s_1}{\max f_1(\Psi(t), \mathbf{P}_i)} + \frac{(1-w)\, s_2}{f_2(\Psi(t))} \quad \forall \mathbf{P}_i : C_{vis}(\Psi(t), \mathbf{P}_i) = 1, \quad (7)$$

Such value describes the aptitude of the aggregate solution obtained at generation t. s_1, s_2 and w are scale factors that control the balance between f_1 and f_2. Note that here we also have $f_1(\Psi(t), P_i) > 0$. On the other hand, aggregate networks which do not provide complete object coverage are penalized with $f_{global}(\Psi) = 1$. Obviously the goal of the algorithm is to encourage the improvement of this global fitness along the course of successive generations. However, another purpose of such evaluation is to be able to reflect on the population of partial solutions the effect of the evolutionary process along the aggregate solutions. Individuals participating of the aggregate solution will be rewarded or penalized based on its global fitness; as well as, on the complete solution characteristics. On the other hand, promising individuals not selected should be compensated so they might contribute in latter stages of the evolutionary process. In this way, the process of redistributing the global fitness plays a key factor in the strategy of individual evolution.

Global Fitness Redistribution

The best solution to the network design problem is one that reconstructs completely the object with the highest accuracy. Under the Parisian approach this solution is created by aggregating multiple individuals from an evolving population. In this way, individuals that contribute to attaining and improving valid solutions should be favored in the evolutionary process. This requires addressing aspects of function optimization and constraint satisfaction. In this subsection we shall describe how the global fitness evaluation is used to deal concurrently with both of these issues. Our approach consists in periodically adjusting the local fitness values of individuals in the population based on the results of global fitness evaluations. In particular, the local fitness value of a single individual is incremented or decremented after considering aspects such as: global fitness value of the aggregate solution, local fitness values of other individuals, as well as the individual's potential for improving the aggregate solution. This is achieved by defining two different local fitness adjustment functions, one to promote global fitness optimization and another to promote global constraint satisfaction.

Function optimization will be addressed first. In order to reflect the quality of an aggregate solution $\Psi(t)$ on each of the individuals ψ^j that compose it, we

use the ratio of improvement in global fitness among successive generations. The magnitude of the adjustment of an individual's local fitness is proportional to this ratio as follows:

$$g_1(\psi^j) = f_{loc}(\psi^j)\left[\frac{f_{global}(\Psi(t))}{f_{global}(\Psi(t-1))} - 1\right] \qquad \forall \psi^j \in \Psi(t). \tag{8}$$

The fitness of each of the individuals composing the solution $\Psi(t)$ will be multiplied by this ratio, enhancing or degrading its individual local fitness accordingly. Now we shall consider the constraint satisfaction. It is very likely that each individual subnetwork will only cover part of the object. It is also possible that a given aggregation of individuals will not provide full object coverage. In this respect, when a particular aggregate solution $\Psi(t)$ does not cover some object region $\mathbf{P}i$ (e.g., $C_{vis}(\Psi(t), \mathbf{P}_i) = 0$) it would be desirable to enhance the fitness value of those individuals on the population that indeed cover such region. The amount of enhancement of those individuals shall be proportional to their difference in fitness with respect to the best individual in the population. Hence, we have

$$g_2(\psi^j) = f_{local}(\psi^{best}) - f_{local}(\psi^j) \qquad \forall \psi^j : \mathbf{V}(\psi^j) \bigcap \overline{\mathbf{V}(\Psi(t))} \neq \emptyset. \tag{9}$$

Note that this value is only calculated for those individuals which cover an object region not sensed by the aggregate solution computed during the current generation. In other words, it is calculated by those individuals satisfying global constraints that are not satisfied by the aggregate solution. Once both fitness adjustment functions are calculated, the global fitness is fed-back to the general population as follows:

$$f_{local}(\psi^j) = \begin{cases} f_{local}(\psi^j) + \lambda_1 g_1(\psi^j) & \text{if } \psi^j \in \Psi(t) \\ f_{local}(\psi^j) + \lambda_2 g_2(\psi^j) & \text{if } \mathbf{V}(\psi^j) \bigcap \overline{\mathbf{V}(\Psi(t))} \neq \emptyset \\ f_{local}(\psi^j) & \text{otherwise .} \end{cases}$$

Here, λ_1 and λ_2 are user defined parameters that reflect the relative importance given to each of the aspects involved in the global fitness redistribution.

Population Diversity Preservation

Maintaining a diverse set of individual solutions is a prerequisite for our implementation of the Parisian approach. This is relevant because our search for an optimal configuration is developed over a highly multi-modal space. In evolutionary computing diversity preservation techniques can generally fall into some of the following categories: (a) heuristic modification of genetic operators in order to promote diversity, (b) fitness function penalization for crowded individuals, and (c) incorporation of some higher level algorithmic structure to generate and

manage sub-populations. In this work, the fitness sharing scheme is adopted [9]. In this way, the fitness of an individual is adjusted by

$$f'_{local}(\psi_j) = \frac{f_{local}(\psi_j)}{\sum_{i=1}^{K} sh(\psi_i, \psi_j)}, \quad where$$

$$sh(\psi_i, \psi_j) = \begin{cases} 1 - \frac{\|\psi_i, \psi_j\|}{\sigma_{sh}} & \text{if } \|\psi_i, \psi_j\| < \sigma_{sh} \\ 0 & \text{otherwise .} \end{cases}$$

Since our individuals represent sets of spatially distributed cameras, the chosen metric was the Hausdorff distance. This metric is defined for our problem by the following expression

$$\|\psi_i, \psi_j\| = h(\psi_i, \psi_j) = \max_{a \in \psi_i} \{ \min_{b \in \psi_j} \{ d(a, b) \} \},$$

where a, b represent the 3D positions of each camera in a given network and $d(a, b)$ is the Euclidean distance among points. Geometrically, this metric expresses the maximum distance of a set to the nearest point in the other set. Based on such geometrical interpretation we can empirically define an appropriate sharing radius σ_{sh}. A related issue is the choice of a selection operator during evolution. One choice could be using either ranking based selection or fitness proportional selection. It has been reported that tournament selection, is not adequate for fitness sharing approaches to multi-modal optimization due to the high selection pressure [14]. Hence, in our approach we use a stochastic remainder selection operator. This choice is also justified by the fact that we use proportional fitness adjustment during global fitness redistribution.

Aggregation of Individuals

At each generation a set of individuals is selected from the population in order to form a composite solution by means of aggregation. The approach by which such selection is carried out reflects directly on the quality of the solutions obtained by our algorithm. Such procedure can be viewed as taking a sample of individuals from the population. A brute force approach which evaluates every possible combination of individuals is discarded from consideration due to its computational cost. An alternative would be that of having a procedural mechanism by which a composite solution is incrementally constructed from the available population. This is in principle similar to incorporating local search in evolutionary techniques (e.g., memetic algorithms) and is necessarily application dependent. A more general approach is to use the individual's fitness values to influence the aggregation mechanism. In such a case we have the choice of either deterministic (i.e., elitist) or stochastic (i.e., roulette, tournament) selection procedures. In order to make such decision, the characteristics of the diversity preservation method must be taken into account. Fitness sharing mechanisms attempt to form and maintain clusters of individuals over each of the multi-modal function local maxima (or minima as the case may be). The number of elements in each

of these maxima should be proportional to its magnitude. Hence, the selection of the best J individuals from the population is likely to produce very similar individuals belonging to a cluster, even for a well distributed population. Under such scenario, a clustering technique would be desirable in order to properly identify each local maxima for consideration into the aggregate solution [3]. To avoid such calculations we have implemented the following simple procedure:

```
for i=1 to J
   - Select the Best Individual in the Population
   - Eliminate all Individuals within the Sharing Radius
```

Naturally, special considerations need to be taken for the case where the population does not provide enough diversity for selecting J different individuals. However, this procedure takes advantage of the fact that distances among individuals have already been calculated in the fitness sharing stage. Heuristic provisions can be made to ensure that the selected individuals are maintained across several generations. This would correspond to elitist selection in typical genetic algorithms.

4 Experimental Results

The reconstruction of a complex 3D object is considered in our experimentation. The goal is to determine a viewing configuration that will offer optimal results in terms of reconstruction accuracy. Here, we shall consider the design of a fixed size camera network of $M = 9$ stations. According to our approach, the level of granularity of our problem decomposition needs to be established. For these series of experiments we will use camera subnetworks of $N = 3$ cameras. In this way, each of the individuals in the population will consist of a vector $\psi \in \mathbb{R}^6$. Hence, a total of $J = 3$ subnetworks will need to be aggregated in order to form a complete solution to our network design problem. The convex polyhedral object under study, depicted in Fig. 3, is partitioned into $R = 6$ regions. The selection of individuals for solution aggregation is based on their fitness value. Finally, the user defined valued λ_1 and λ_2 are set to $\lambda_1 = \lambda_2 = 1.0$. For all our experiments, SBX-crossover is utilized with a probability $Pc = 0.95$ along with polynomial mutation subject to $Pm = 0.05$. A sharing radius of $\sigma_{sh} = 0.75$ was applied.

4.1 Algorithm Performance

We have used stochastic remainder selection for reproduction under generational replacement. At the same time, the same global fitness function was optimized by a typical genetic algorithm (e.g., each individual encodes a complete solution). This was done in order to have some reference point in the assessment of our proposed methodology. Using a population of 30 individuals, both evolutionary algorithms were executed for 100 generations.

Fig. 4 plots population performance measures (best, mean, and worse fitness) for a canonical GA on the left and also for our Parisian approach on the

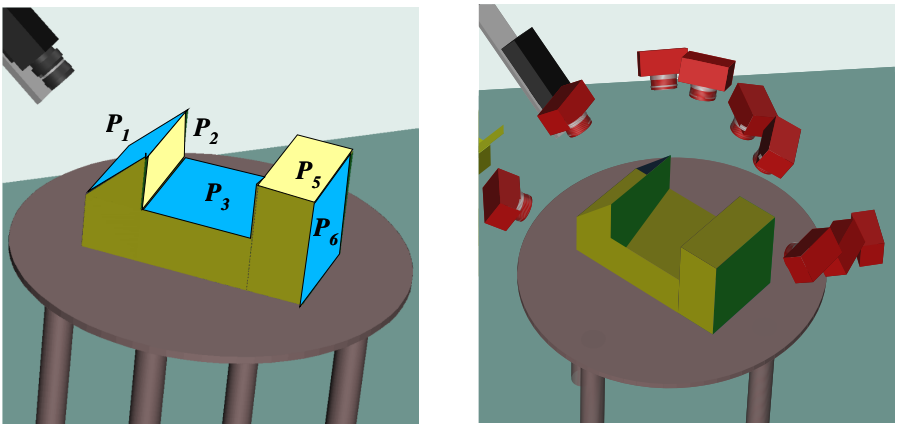


Fig. 3. The 3D object under observation. The concave object is partitioned into different regions in order to facilitate the fitness evaluation of subnetworks of small size. A photogrammetric network formed by nine cameras is illustrated on the right.

right. While these measures are descriptive of the dynamics of our population, the importance is on the aggregate solution fitness measure. In this respect, our approach slightly outperforms a canonical methodology in terms of solution quality. However, these results are made more relevant when considering the computational cost involved in fitness evaluation. For our studied object, evaluation based on criterion (1) of a complete network of nine cameras is over 15 times more costly than that of a three camera subnetwork. Accordingly, by virtue of our problem decomposition, the total execution time of the algorithm is reduced 10 times. Clearly, a significant benefit in performance has been achieved.

4.2 Imaging Geometry Configurations

Analysis and comparison among the best configurations found by our algorithm are comparable with the geometrical distributions reported in [15]. Fig. 5 depicts two of the configurations found by our algorithm. Note the geometrical similarities among both configurations. Each of these networks is composed of three subnetworks that are integrated by three cameras. Different color schemes depict the membership of each particular camera to a given subnetwork. These subnetworks correspond to a single individual in the population. In both configurations some of the cameras present mixed colors, indicating the composition of at least two cameras located in the same position. This corresponds to the Second Order Design stage in photogrammetric network design. The Parisian approach found similar geometrically configurations from an aggregation of distinct individuals in both cases. Nevertheless, the configuration on the right has an improvement of 4.2% on the fitness value in terms of precision. Such discrepancies illustrate the high non-linearity of our search space.

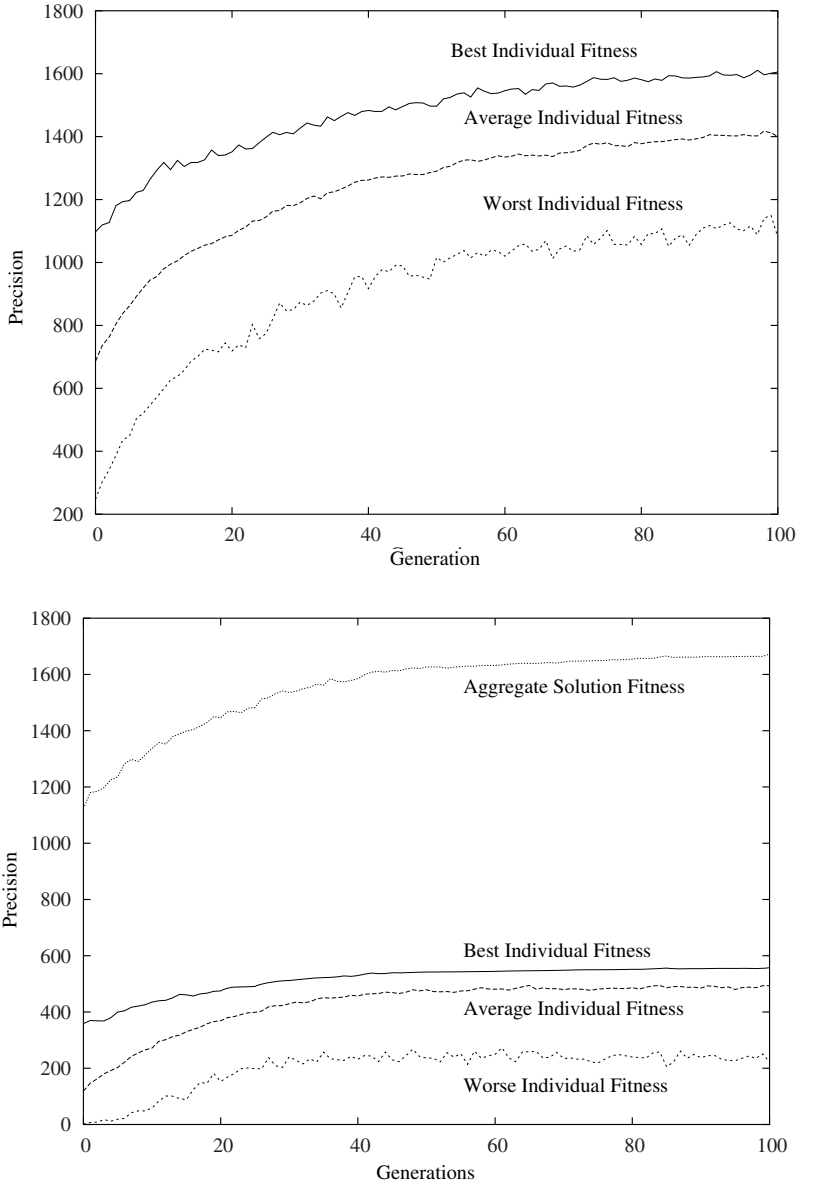


Fig. 4. Performance comparison. On the left, the population evolution of a typical genetic algorithm is depicted. On the right, higher fitness values are consistently attained by the aggregate solutions of our proposed methodology. Plotted values reflect the averages over 20 executions with $\lambda_1 = \lambda_2 = 1.0$.

4.3 Parameter Setting

The choice of mixing values λ_1, λ_2 is an important aspect in the performance of the algorithm, as they determine the magnitude of the global fitness adjustment given to each individual. In order to exemplify such phenomena we have carried out different experiments varying the ratio and magnitude of these values. Experiments show a fairly robust behavior for similarly scaled values under 1.0. In general, performance deteriorates with respect to an increment on these parameters. The right plot of Fig. 6 illustrates the scenario where constraint satisfaction is completely predominant over function optimization. As a result, the fitness value of aggregate solutions is decreased by weaker configurations that are unreasonably enhanced by the global fitness evaluations.

4.4 Problem Decomposition Granularity

The level of granularity also plays a major role in the performance of our system. This is reflected in the quality of our solutions, as well as on the efficiency of our system. In order to illustrate it, the case of a 12 camera network is studied. Experiments for sub-networks of 2, 3, 4 and 6 cameras were carried out and results compared against a canonical evolutionary algorithm. For all experiments a population of 50 individuals was used. Also, mixing parameters were set to $\lambda_1 = \lambda_2 = 1.0$ and the sharing radius set to $\sigma_{sh} = 0.75$. The performance results after 20 executions are presented in Table 1. The most accurate configuration was obtained by the canonical evolutionary approach. Among the results for different levels of decomposition by our Parisian approach, the results favor the choice of individual subnetworks of small size, as they give better fitness values along with the highest computational speed-up. The geometric disposition of

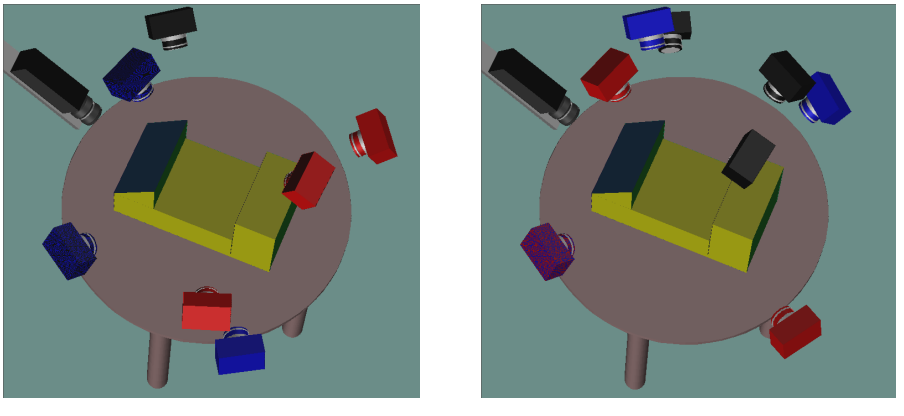


Fig. 5. These images show different imaging geometries obtained by our approach, which represents the best solutions found at different executions of our algorithm. Membership to a given subnetwork is depicted by camera color.

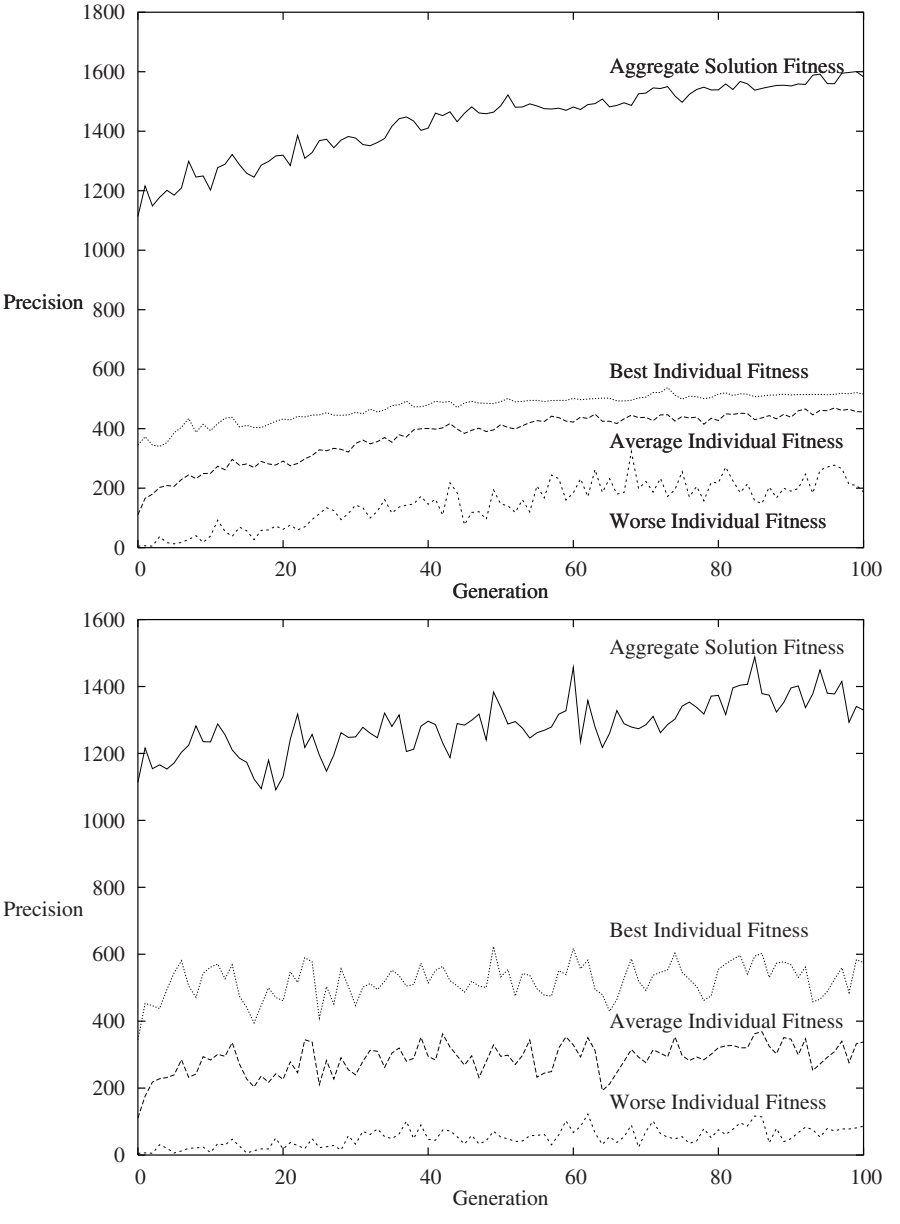
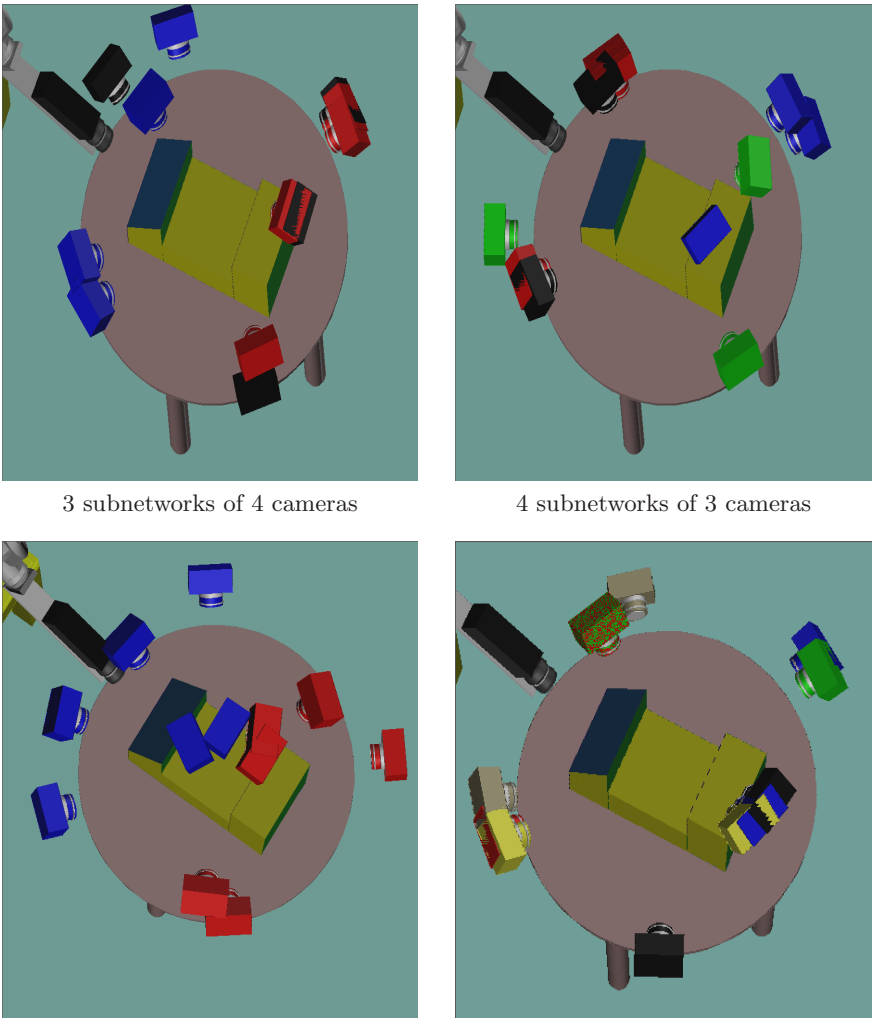


Fig. 6. Dependence on parameters λ_1, λ_2. The plot on the left corresponds to an execution with mixing values $[\lambda_1 = 0.8, \lambda_2 = 0.2]$ Performance is slightly deteriorated. The plot on the right represents an execution with values $[\lambda_1 = 0, \lambda_2 = 1.5]$, which gives almost random algorithm performance.



3 subnetworks of 4 cameras

4 subnetworks of 3 cameras

2 subnetworks of 6 cameras

6 subnetworks of 2 cameras

Fig. 7. Network configurations of 12 cameras with different levels of decomposition granularity

these resulting camera networks is depicted in Fig. 7. While, in this scenario, the Parisian approach attained slightly lower fitness values than a canonical approach. Note that an almost 30 times reduction in execution time was achieved. Again, a significant benefit in terms of performance has been achieved.

Table 1. Results after 20 executions of our algorithm for different levels of granularity

Subnetwork size	Best fitness	Computational speed-up
2	1666.43	29.72
3	1630.21	21.41
4	1310.84	11.48
6	1758.68	5.03
12	1702.05	1.0

5 Conclusion

The Parisian approach to evolutionary computation offers an efficient way to address the problem of automated camera placement, while preserving the validity of photogrammetric procedures. In fact, by virtue of an adequate problem partition and decomposition, solution quality is improved with considerable reductions in computational effort for the considered scenarios. Future work includes incorporating rigorous bundle adjustment procedures, where the computational savings should be even more dramatic. However, such research lines require careful considerations regarding Zero Order Design for photogrammetric networks, due to the need for a common datum in bundle adjustment procedures. This work has developed an efficient optimization technique based on an original conception of population based evolutionary optimum seeking. In particular, a novel application for the Parisian approach has been described and important application related aspects have been addressed. This work incorporated canonical evolutionary principals in order to achieve the goal of evolving a solution based on the evolution of its components. While promising experimental results are obtained, a lack of theoretical principals describing the algorithm behavior is still pending as in general evolutionary algorithms. Nevertheless, there are many specialized evolutionary approaches that could be used in conjunction with our proposed methodology, such as parallel evolutionary algorithms, other co-evolution techniques or even multi-objective evolutionary algorithms.

Acknowledgement. This research was funded by CONACyT and INRIA through the LAFMI project 634-212. Second author supported by scholarship 142987 from CONACyT.

References

1. Chen, S.Y., Li, Y.F.: Automatic sensor placement for model-based robot vision. IEEE Trans. Syst., Man Cybernet., Part B 34(1), 393–408 (2004)
2. Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Individual GP: an alternative viewpoint for the resolution of complex problems. In: Banxhaf, E., Daida, J., Eiben, A.E., Garzon, M.H., Honovar, V., Jakiela, M., Smith, R.E. (eds.) Genetic and Evolutionary Computation Conf. GECCO 1999. Morgan Kaufmann, San Francisco (1999)

3. Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Polar IFS + Parisian Genetic Programming = Efficient IFS Inverse Problem Solving. Genet. Programm. Evolvable Mach. J. 1(4), 339–361 (2000)
4. Dunn, E., Olague, G., Lutton, E.: Automated Photogrammetric Network Design Using the Parisian Approach. 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. 356–365. Springer, Heidelberg (2005)
5. Firoozfam, P., Negahdaripour, S.: Theoretical Accuracy Analysis of N-Ocular Vision Systems for Scene Reconstruction, Motion Estimation, and Positioning. In: 2nd Internat. Symp. on 3D Data Processing, Visualization, and Transmission (3DPVT 2004), September 2004, pp. 888–895 (2004)
6. Fraser, C.S.: Limiting Error Propagation in Network Design. Photogramm. Eng. Remote Sens. 53(5), 487–493 (1987)
7. Fraser, C.S.: Network Design. In: Atkinson, K.B. (ed.) Close Range Photogrammetry and Machine Vision, pp. 256–281. Whittles Publishing, Caithness, Scotland (1996)
8. Fraser, C.S., Woods, A., Brizzi, D.: Hyper Redundancy for Accuracy Enhancement in Automated Close Range Photogrammetry. Photogramm. Record 20(111), 205–217 (2005)
9. Goldberg, D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. Genetic algorithms and their applications. In: Proc. 2nd Internat. Conf. on Genetic Algorithms, pp. 41–49 (1987)
10. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
11. Hörster, E., Lienhart, R.: On the Optimal Placement of Multiple Visual Sensors. Internat. Multimedia Conference. In: Proc. 4th ACM internat. Workshop on Video Surveillance and Sensor Networks, pp. 111–120 (2006)
12. Mason, S.: Heuristic Reasoning Strategy for Automated Sensor Placement. Photogramm. Eng. Remote Sens. 63(9), 1093–1102 (1997)
13. McGlone, C. (ed.): Manual of Photogrammetry. American Society for Photogrammetry and Remote Sensing, Bethesda, MD, p. 1151 (2004)
14. Oei, C., Goldberg, D., Chang, S.: Tournament Selection. Niching and the Preservation of Diversit. IlliGAL Report No. 91011. Urbana, IL, university of Illinois at Urbana-Champaign (1991)
15. Olague, G.: Automated Photogrammetric Network Design Using Genetic Algorithms. Photogramm. Eng. Remote Sens. 68(5), 423–431 (2002); Awarded "2003 First Honorable Mention for the Talbert Abrams Award", by ASPRS
16. Olague, G., Mohr, R.: Optimal Camera Placement for Accurate Reconstruction. Pattern Recognition 35(4), 927–944 (2002)
17. Olague, G., Dunn, E.: Development of a Practical Photogrammetric Network Design using Evolutionary Computing. Photogramm. Record 22(117), 22–38 (2007)
18. Saadatseresht, M., Fraser, C., Samadzadegan, F., Azizi, A.: Visibility Analysis In Vision Metrology Network Design. Photogramm. Record 19(107), 219–236 (2004)
19. Saadatseresht, M., Samadzadegan, F., Azizi, A.: Automatic Camera Placement in Vision Metrology Based On A Fuzzy Inference System. Photogramm. Eng. Remote Sens. 71(12), 1375–1386 (2005)

20. Tsai, M.J., Hung, C.C.: A Fast Evaluation Approach of Geometrical Correspondence Uncertainty for 3-D Vision Measurement System. JSME International Journal Series C 49(2), 527–534 (2005)
21. Wong, C., Kamel, M.: Comparing Viewpoint Evaluation Functions for Model-Based Inspectional Coverage. In: 1st Canadian Conf. on Computer and Robot Vision (CRV 2004), May 2004, pp. 287–294 (2004)

Adaptive Estimation of Distribution Algorithms

Roberto Santana[1], Pedro Larrañaga[1], and José A. Lozano[1]

Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country
Paseo Manuel de Lardizabal 1, 20080 Donostia - San Sebastian, Spain
rsantana@si.ehu.es, pedro.larranaga@ehu.es, ja.lozano@ehu.es

Summary. Estimation of distribution algorithms (EDAs) are evolutionary methods that use probabilistic models instead of genetic operators to lead the search. Most of current proposals on EDAs do not incorporate adaptive techniques. Usually, the class of probabilistic model employed as well as the learning and sampling methods are static. In this paper, we present a general framework for introducing adaptation in EDAs. This framework allows the possibility of changing the class of probabilistic models during the evolution. We present a number of measures, and techniques that can be used to evaluate the effect of the EDA components in order to design adaptive EDAs. As a case of study we present an adaptive EDA that combines different classes of probabilistic models and sampling methods. The algorithm is evaluated in the solution of the satisfiability problem.

Keywords: Estimation of distribution algorithm, adaptive probabilistic model, SAT.

1 Introduction

Estimation of distribution algorithms (EDAs) [9] are evolutionary methods that use probabilistic models to represent relevant information about the search space. The idea is to capture, in the form of probabilistic dependencies between the variables, information about promising areas of the search space that can be used to improve the search for better solutions. Machine learning techniques are used to learn the probabilistic models and sample new solutions from them. EDAs have shown to solve problems where genetic algorithms exhibit a poor performance [9,12].

A characteristic feature of EDAs is the type of probabilistic model used. Different models come associated with different capacities of representation and the computational complexity of the algorithms used to learn and sample from them also changes accordingly. Although probabilistic models provide EDAs with an important degree of flexibility, usually the class of the models is fixed at the beginning of the evolution and will not change during the search process. This fact may compromise the flexibility of the algorithm. More efficient EDAs are

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 177–197, 2008.
springerlink.com

expected to exhibit a wider amount of adaptation with more flexible frameworks for probabilistic modeling.

In this chapter, we present our initial results on the conception of adaptive EDAs. We identify a number of ways in which adaptation can be added to EDAs and focus on the use of adaptive probabilistic models and sample algorithms. Our findings lead to the introduction of an EDA that uses a combination of probabilistic models and which is evaluated in the optimization of a number of instances of the satisfiability problem.

The chapter is organized as follows. In the next section, EDAs are presented and some of their main characteristics are discussed. In Section 3, we accomplish a brief review of previous work on the design of adaptive genetic algorithms. Section 4 introduces a general framework for the analysis and design of adaptive EDAs. In Section 5 we focus on the analysis of feasible ways of incorporating adaptive probabilistic models to EDAs. Section 6 presents factor graph based factorizations and Kikuchi approximations as our case of study. The design of the experiments and the numerical results are presented in Section 7. The paper ends with Section 8 where the conclusions of our paper are presented.

2 Estimation of Distribution Algorithms

2.1 Notation

Let X be a random variable. A value of X is denoted x. $\mathbf{X} = (X_1, \ldots, X_n)$ will denote a vector of random variables. We will use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. S will denote a set of indices in $N = \{1, \ldots, n\}$, and $\mathbf{X}_S$ (respectively, $\mathbf{x}_S$) a subset of the variables of $\mathbf{X}$ (respectively, a subset of values of $\mathbf{x}$) determined by the indices in S. We will work with discrete variables.

The joint probability mass function of $\mathbf{x}$ is represented as $p(\mathbf{X} = \mathbf{x})$ or $p(\mathbf{x})$. $p(\mathbf{x}_S)$ will denote the marginal probability distribution for $\mathbf{X}_S$. We use $p(X_i = x_i \mid X_j = x_j)$ or, in a simplified form, $p(x_i \mid x_j)$, to denote the conditional probability distribution of X_i given $X_j = x_j$.

A *graphical model* for $\boldsymbol{X} = (X_1, \ldots, X_n)$ encodes a graphical factorization of a joint probability distribution. Commonly used graphical models include Bayesian networks, Markov networks and factor graphs.

2.2 EDAs

Estimation of distribution algorithms (EDAs) [9,13] are evolutionary algorithms that work with a set (or population) of points. Initially, a random sample of points is generated. These points are evaluated using the objective function, and a subset of points is selected based on this evaluation. Hence, points with better function values have a higher chance to be selected. Then a probabilistic model of the selected solutions is built, and a new set of points is sampled from the model. The process is iterated until an optimum has been found or another termination criterion is fulfilled.

Algorithm 1. Estimation of distribution algorithm

```
1  Set t ⇐ 0. Generate M points randomly.
2  do {
3      Evaluate the points using the fitness function.
4      Select a set S of N ≤ M points according to a selection method.
5      Calculate a probabilistic model of S.
6      Generate M new points sampling from the distribution
       represented in the model.
7      t ⇐ t + 1
8  } until Termination criteria are met.
```

The general scheme of the EDA approach is shown in Algorithm 1. There are a number of selection methods that can be used. In the literature, truncation, Boltzmann, and tournament selection are commonly used with EDAs.

One essential assumption of these algorithms is that it is possible to build a probabilistic model of the search space that can be used to guide the search for the optimum. Thus, a key characteristic and crucial step of EDAs is the construction of this probabilistic model. If there is available information about the function (e.g. variable dependencies), this can be exploited by including parametrical and/or structural prior information in the model. Otherwise, the model is learned exclusively using the selected population. Several probabilistic models with different expressive power and complexity can be applied. These models may differ in the order and number of the probabilistic dependencies that they represent.

3 Work on Adaptive Genetic Algorithms

In this section, we make a short review of previous work on the design of adaptive genetic algorithms, emphasizing some of the issues that will be considered in the presentation of our proposal of adaptive EDAs.

In [24], an analysis of the way adaptation has been used in genetic algorithms (GAs) is done. Three principles that allow to study the role of adaptation are presented. These are:

- What is being adapted? (operators, parameters, etc.).
- The scope of the adaptation (i.e. does it apply to all the population, just to individual members, or just sub-components?).
- Basis for change (externally imposed schedule, fuzzy logic, etc).

One of the most important benefits of adaptive reproductive operators is that they permit a more flexible tuning between the goals of exploration and exploitation of the search space. This can be done by modifying the parameters associated to the operators, or by changing their frequency of application. The simplest adaptive GAs use a fixed set of operators and adapt the probability

of application of those operators. Another class of adaptive GAs change the behavior of the operators over time.

Different techniques have been used to extract information from the search in order to determine adapting schedules of decision rules for adaptive GAs. This includes the use of fuzzy logic [6], inductive learning [23], and reinforcement learning [16].

There are important similarities between the study of adaptation in GAs and EDAs. Research done on those components common to GAs and EDAs can be applied to the second class of algorithms with minor modifications. This research comprises, for instance, the use of variable population sizes, adaptive selection schedules, etc. The techniques employed to extract information about the search are also of direct application to the conception of adaptive EDAs.

The study of adaptation in EDAs must take into account some main differences between EDAs and GAs. One of these differences is that reproduction, as implemented in GAs, can provide more fine grained information about the effect of the reproduction operator that the way reproduction is accomplished in EDAs. For instance, the concept of "safety ratios" [22] refers to the probability that a new point generated by the application of reproductive operators would be fitter than its parent(s). This probability, that can be used as a measure of the operators effect, is calculated using information about the fitness of the parents. Since the influence of the parents in EDAs (the whole selected set) is mediated by the existence of a probability model, it is not possible to define a parent-to-offspring correspondence. Instead, macroscopic measures (e.g. average fitness of the population) must be used to describe the effect of the reproductive operators.

But even if detailed information about the relationship parent-offsprings will not be available in EDAs, these algorithms expand both, the sources of statistical information about the search, and the range and variety of applications of this information. Probabilistic models are the main specific source of statistical information in EDAs, but also the information collected during the learning of these models (the model search step) could be used for adaptation.

4 Improving the Search: Adaptive EDAs

Our initial analysis will be led to the identification of the particular features of EDAs that should be modified for the design of adaptive algorithms.

There are a number of issues that need to be addressed in order to accomplish the conception of adaptive EDAs. The following questions will help us to guide our analysis:

- Which EDA components can be adapted to the search?
- How can an EDA adapt its components and parameters by itself?
- How to obtain relevant information from the search for adaptation?
- Which is the available repertoire of possibilities for adapting EDAs?

Among the components of EDAs that can be adaptively modified during the search are the fitness evaluation, the selection method, the elistism method and

the parameters (i.e. population size, selection and elitism parameters). Even if research focused on these components is important we will consider in our analysis those other components that are specific to EDAs. These include: the class of probabilistic models, the methods used for learning them and the sampling methods. In Section 5, we will analyze how these components can be modified in order to guarantee an adaptive behavior of EDAs.

Once the parts of the algorithms that can adapt to changes during the search have been identified, it is important to define the ways they can be adapted to the search and which the requirements to implement the changes are. Two essential issues have to be considered for the definition of such strategies. They are: the information about the search history and the general decision rules based on this information.

The first issue involves the collection, storing and interpretation of the data generated during the search. Not all the data available from the algorithm behavior is relevant to the purpose of taking decisions about the search strategies. Therefore, it is needed to set the sort of data that will be stored and eventually used by the algorithm. Furthermore, some data may require a preprocessing step before being used. The computational cost of this step should be estimated in order to guarantee that the gain due to increasing the algorithm adaptability is not achieved at the expense of an unbearable computational cost of preprocessing step.

The second issue, which is very related with information about the search history, it the strategy conceived for using this information. As in the case of adaptive GAs, this strategy can be defined using different machine learning techniques that will employ a particular class of the information available. The selection of the relevant information for adaptive EDAs presupposes that a strategy that will use this class of information has been determined.

We describe in some detail which the possible sources of information for adaptive EDAs and the preprocessing steps needed to use this information are. Main sources of information are the following:

- Fitness related measures:
  1. Measures of convergence.
  2. Measures of exploration and exploitation.
- Information about the interactions captured in the graphical model.

Fitness related measures are obtained from the fitness values of the solutions so far visited by the algorithm. Let $f(t)$ be the fitness function at generation t. Examples of these measures include:

- Average fitness and variance of the population ($\bar{f}(t)$,$\sigma^2(f(t))$).
- Response to selection: $R(t) = \bar{f}(t+1) - \bar{f}(t)$.
- Amount of selection: $S(t) = \bar{f}_s(t+1) - \bar{f}(t+1)$.
- Realized heritability $b(t) = \frac{R(t)}{S(t)}$.

The average fitness is used to compute the response to selection which is a general measure of the improvements obtained in the average fitness of the

population by the application of the learning and sampling steps. However, an increase of $R(t)$ can hide a loss of diversity in the population. In these cases, the change in the fitness variance can support additional information about whether the population is really diverse. The mathematical framework that involves the use of $R(t)$, $S(t)$ and $b(t)$ was originally proposed in population genetics and has been applied before to the analysis of the breeder genetic algorithm [14]. We propose to apply these measures to evaluate the role of different operators (e.g. different classes of probabilistic models and sampling methods) and parameters used by EDAs.

Other measures that can be used as a source of information about the search are the average fitness of individuals in the selected population, the number of different solutions in the selected population and the number of generations spent without improvement.

Among the measures related to the probabilistic model that can be used for adaptation are: the number of edges, the size of the maximum clique, the number of maximal cliques, and the number of connected components of the graph.

Alternatives for adaptation in EDAs include the following:

- Varying the strength of selection according to the diversity of solutions.
- Choice of the probability model class according to the graph complexity.
- Determination of the sampling algorithm according to the graph topology.
- Increasing the population size to avoid stagnation of the search.

In [10], an adaptive schedule for the Boltzmann selection was introduced and compared with the truncation selection. Although both methods showed similar dynamics, EDAs with truncation selection reached better convergence rates and required less number of fitness evaluations. In [17], adaptive priors that relate the rate of variation of the population to the quality of the approximation learned by the model are proposed in the context of the mixture of trees factorized learning algorithm (MT-FDA) [21]. Better results than when using MT-FDA with static learning methods are achieved.

There is some recent work on the incorporation of adaptive techniques in EDAs [2,5,26]. This work has focused on optimization problems with continuous representation and the mechanisms of adaptation have been constrained to the change in the parameters governing the learning process for the probabilistic model of choice. Although some of the general issues we treat in this paper can be extended to problems with continuous representation, our proposal is introduced in the context of optimization problems with discrete representation. On the other hand, the scheme of adaptation we present allows to change the class of the probabilistic models during the evolution, expanding the class of components and the scope of actions available to deal with the exploration of the search space.

In the next section, we will focus on the definition of a framework that allows to change the class of probabilistic model and the sampling algorithm during the search.

5 Adapting the Class of Probabilistic Models in EDAs

In order to explain the ways adaptation can be introduced in EDAs we start by presenting a generalized EDA that comprises different types of probabilistic models, learning and sampling algorithms. We constrain our analysis to EDAs based on undirected graphical models [15,19,20]. The pseudocode of the generalized EDA is shown in Algorithm 2.

Algorithm 2. Generalized EDA

```
1  Set t ⇐ 0. Generate M points randomly.
2  do {
3      Select a set S of N ≤ M points according to a selection method.
4      Learn an undirected-graph-based representation of the dependencies
       in S.
5      Using the graph, determine a class of graphical models or
       approximation strategy to approximate the distribution
       of points in S.
6      Determine an inference algorithm to be applied in the graphical
       model.
7      Generate M new points from the model using the inference method.
8      t ⇐ t + 1
9  } until Termination criteria are met.
```

The most relevant feature of the generalized EDA is that it allows the use of different classes of graphical models at each generation. The model choice should be related to the complexity of the data and to the patterns of interaction between the components of the problem. In situations in which there are few interactions between the variables, we could choose a simple class of models and avoid more complex learning algorithms (e.g. those required by Bayesian networks). Choosing a simpler model can thus lead to an advantage in terms of computational time. Additionally the marginal probabilities of a probabilistic model with lower order dependencies could be more accurately estimated from small data samples.

Using different classes of graphical models during the search will also allow to incorporate different sampling techniques that determine different ways of searching for solutions. Therefore, the dynamic change of the probabilistic model will need an automatic procedure to select among the different types of graphical models. The topological characteristics of the undirected graphs learned are plausible information for this decision. The number, size, and cardinality of the variables (number of values) of each clique are three of the issues that influence the feasibility of the model for estimating the marginal probabilities and sampling new solutions.

We will assume that an undirected graph that encodes the (in)dependence relationships between the variables is given. Given the structure, we face two

problems: 1) To decide which candidate probabilistic models could be used as approximations, and 2) To define which criteria to take into account to choose among them.

5.1 Alternatives for Probabilistic Modeling

Table 1 shows a number of alternatives for selecting a probability model according to the graph structure. Column 1 (Graphs) describes whether the approximation comprises all and only those dependencies in the graph (exact), a subgroup of the dependencies (subgraph) or all the dependencies and additional dependencies (triangulated graph). Column 2 (Graphical models) describes different situations that could be faced (e.g. univariate –there are not dependencies–, junction tree –valid factorization–, etc.). Column 3 (Inference) shows different sampling algorithms that can be used according to the model. They comprise: probabilistic logic sampling (PLS), Gibbs sampler (GS), most probable configurations (MPC), most probable configurations with belief propagation (MPC-BP), most probable configurations with loopy belief propagation (MPC-loopy BP), and most probable configurations with generalized belief propagation (MPC-generalized BP).

Table 1. Approximation strategies, graphical models, and inference methods to be employed by EDAs based on undirected graphs

Graphs	Graphical models	Inference
exact graph	univariate	PLS,MPC
	junction tree	PLS,MPC-BP
	junction graph	PLS,MPC-BP
	clique-based Kikuchi approximation	GS
	Bethe approximation	MPC-loopy BP
	Kikuchi approximation	MPC-generalized BP
subgraph	univariate	PLS,MPC
	junction tree	PLS,MPC-BP
	junction graph	PLS,MPC-BP
	clique-based Kikuchi approximation	GS
	Bethe approximation	MPC-loopy BP
	Kikuchi approximation	MPC-generalized BP
triangulated graph	junction tree	PLS,MPC-BP

5.2 Decision Criteria for Choosing the Model

The second question is the definition of the decision criteria for selecting among the alternatives. Without considering information about the search state, something that will be required for adaptive EDAs, the two main criteria to take into account are the accuracy and the complexity of the approximation. The accuracy of the approximation can be roughly estimated by measuring the number of interactions of the original graph covered by the approximation and the strength of the interactions covered.

Complexity is related to the size of the factors involved in the factorization. One way to choose between the classes of possible approximations according to their complexity is to constrain the size of the largest marginal table as well as the number of factors. To do this, a first step is to calculate all the maximal cliques of the graph and determine the size of the probability tables. To simplify our analysis, we will assume that all the variables have the same cardinality and, therefore, the largest marginal table will correspond to the maximum clique of the graph. The analysis can be generalized to the case where variables have different number of values.

If the graph is triangulated, and the maximum clique of the graph fulfills the complexity constraint, any of the alternatives listed in Table 1 as *exact graph* could be applied. These alternatives respect all the original dependencies that exist in the graph. Nevertheless, the chosen sampling method may determine that only an approximation is achieved.

If the graph is not triangulated, then one possibility is to triangulate it, compute the maximum cliques of the graph, evaluate whether the complexity constraint is fulfilled for the triangulated graph, and in that case, apply any of the alternatives listed in Table 1 as *triangulated graph*.

If the complexity constraint is not fulfilled in the original or in the triangulated graph, then other types of approximation must be tried. One possibility is to simplify the graph by splitting the largest cliques, something that can be done by removing edges. Another possibility is to make the graph sparser in one step previous to the calculation of the cliques.

The most common method applied for inference in the context of EDAs is the PLS. It starts from an order of the variables imposed by the structure of the graphical model. Each variable is sampled given the values assigned to its ascendants in the order. PLS can be applied to the junction tree and junction graph, but it cannot be applied to any other approximation listed in Table 1 because, in the general case it is not possible to find an order of the variables for these approximations.

For Kikuchi approximations that use clique-based decompositions [19], GS can be employed. In this case, the conditional probability distributions serve to determine the transitions in the Markov chain. The drawback of using Gibbs sampling is that if the most probable configuration has an exponentially small probability a large number of configurations will need to be visited to hit the optimum. A partial remedy to this situation is the combination of Kikuchi approximations with propagation based inference methods [7].

6 A Case Study: Generalized Factorized Distribution Algorithms

We will focus now on the class of EDAs that explicitly construct a factorization of the distribution.

6.1 Factorizations

In simple terms, a factorization of a distribution $p(\mathbf{x})$ will be a product of probability distribution $p(\mathbf{x}_s)$ each of which will be called a factor. Factorizations are important because they allow us to obtain a condensed representation of otherwise very difficult to store probability distributions. Generally, graphical models serve to define one or more factorizations of $p(\mathbf{x})$.

The structure of a factorization can be directly recovered from a chordal graph as done in the factorized distribution algorithm (FDA) [12] or learned from data [15,19,20]. Factorizations that satisfy the running intersection property (RIP) are called *valid* [12]. In [19], invalid factorization were further classified in "ordered" and "messy" regarding the number of factors that are part of the factorization. Most of EDAs employ valid factorizations. EDAs that work with messy factorizations were presented in [19,20].

FDA can work with invalid factorizations [12] but in this case, the convergence properties proved for when valid factorizations are employed do not hold. Valid factorizations can also be obtained from directed graphs as those used by EDAs based on Bayesian networks [4].

6.2 Factor Graphs and Factorizations

The analysis of the EDAs presented in this chapter will be based on the use of factor graphs. One convenient way of representing factorizations are factor graphs.

Factor graphs

A *factor graph* [8] is a bipartite graph that can serve to represent the factorized structure of a distribution. It has two types of nodes: variable nodes (which we draw as a circle), and factor nodes (which we draw as a square). In the graphs, factor nodes are named by capital letters starting from A, and variable nodes by numbers starting from 1. We will index variable nodes with letters starting with i, and factor nodes with letters starting with a. The existence of an edge connecting variable node i to factor node a means that x_i is an argument of function f_a in the referred factorization. Figure 1 (left) shows a factor graph with two factor nodes and five variable nodes. The associated undirected graph (right) have two maximal cliques.

In [1], Gibbs distributions are associated with factor graphs. A factor f with scope $\mathbf{X}_S$ is a mapping from $\mathbf{x}_S$ to $\mathcal{R}^+$. A Gibbs distribution $p(\mathbf{x})$ is associated with a set of factors $\{f_a\}_{a=1}^m$ with scopes $\{\mathbf{X}_{S_a}\}_{a=1}^m$, such that

$$p_f(\mathbf{x}) = \frac{1}{Z} \prod_{a=1}^{m} f_a(\mathbf{x}_{S_a}) \tag{1}$$

Factorizations commonly used by EDAs can be represented by factor graphs. For a given function, if its definition sets and the corresponding subfunctions are known then it is possible to associate a factor to each definition set. The

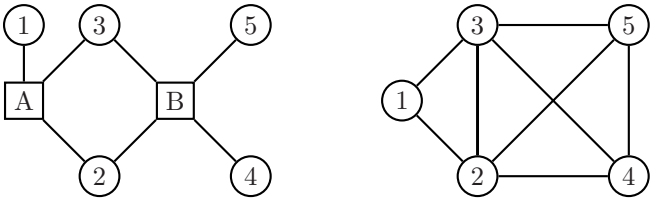


Fig. 1. Factor graph (left) and associated undirected graph with two maximal cliques (right)

corresponding factor graph distribution would be given by (11). At each generation of the EDA, a different factor graph distribution can be learned by taking $f_a(\mathbf{x}_{S_a}) = p_a(\mathbf{x}_{S_a})$ where $p_a(\mathbf{x}_{S_a})$ are the marginal probability distributions learned from the data.

If the factorization is valid then $Z = 1$, and the factorization given by the factor graph is exact. But in the general case, a factorization represented by a factor graph does not have to satisfy the RIP. As a consequence, $Z \neq 1$ and inference of points from the factorization is not straightforward. One alternative in these cases is to learn an approximation. One example of such approximations is the Kikuchi approximation of the distribution.

6.3 Kikuchi Approximation of a Distribution

The Kikuchi approximation of a distribution has three essential components:

1. An initial representation of the interactions of the variables given by a graphical model.
2. A set of regions comprising sets of variables.
3. An overcounting number associated to each region.

In [19], the Kikuchi approximation of a distribution was defined from an independence graph. Initial regions corresponded to the maximal cliques of the graph and the rest of regions were found using the cluster variation method [25]. Overcounting numbers c_R corresponding to each region R were constrained to be calculated using the following recursive formula:

$$c_R = 1 - \sum_{\substack{S \in \mathcal{R} \\ R \subset S}} c_S, \tag{2}$$

where c_S is the overcounting number of any region S in $\mathcal{R}$ such that S is a superset of R. c_R values corresponding to the initial regions are equal to 1.

Given a factor graph, a straightforward generalization of Kikuchi approximations for factor graphs will associate each factor of the graph with an initial region of the Kikuchi approximation. From the set of initial regions the Kikuchi approximation is constructed using the cluster variation method. The overcounting numbers are also calculated using Equation (2).

Given a set of regions $\mathcal{R}$ calculated as explained before, the Kikuchi approximation, $k(\mathbf{x})$, of the probability distribution $p(\mathbf{x})$ is defined as:

$$k(\mathbf{x}) = \prod_{R \in \mathcal{R}} p(\mathbf{x}_R)^{c_R} \tag{3}$$

An important property of the Kikuchi approximation is that, if the factorization is valid, the corresponding Kikuchi approximation is exact, i.e. it will give the original factor graph distribution constructed from the marginal probabilities. On the other hand, a probability function $\tilde{p}(\mathbf{x})$ based on the Kikuchi approximation can be found by normalizing k.

$$\tilde{p}(\mathbf{x}) = \frac{k(\mathbf{x})}{\sum_{x'} k(\mathbf{x}')}$$

Another alternative to deal with factor graph distributions is the use of Markov blanket canonical factorizations [1]. In this case, factor graph distributions are parameterized as a product of local probabilities only. These local probabilities are defined over factor scopes and their Markov blankets [1].

6.4 Learning and Sampling the Kikuchi Approximation from a Factor Graph Distribution

The complexity of learning a Kikuchi approximation from a factor graph distribution is related to whether the structure is previously known, or both the structure and the parameters of the distribution have to be learned. In the first case, and assuming that the maximum size of the factors is feasible regarding the cost of computing and storing the parameters, learning is reduced to estimate the parameters from the data. In the second case, structural learning is required.

The complexity of sampling a factor graph distribution depends on whether the factorization is valid or invalid. In the first case, probabilistic logic sampling could be applied. In second case, more costly techniques like Gibbs sampling [19] and belief propagation [11] could be employed.

To learn the structure of the factor graph we follow the approach described in Algorithm 3.

Algorithm 3. Algorithm for learning a factor graph representation

1. Learn an independence graph G from the data (the selected set of solutions).
2. If necessary, refine the graph.
3. Find the set $\mathcal{C}$ of all the maximal cliques of G.
4. Associate a factor to each maximal clique of the graph.
5. Find the set of regions $\mathcal{R}$.
6. Find the marginal probabilities for the regions.

Given an undirected graph $G = (V, E)$, a clique in G is a subset of V for which there exists an edge between every pair of vertices. A clique is the maximum clique of the graph if it is a clique with the highest number of vertices. The choice of taking maximal cliques as factors is related to the properties of the Kikuchi approximation for clique based decompositions shown in [19].

The independence graph is learned using independence tests. We use the Chi-square independence test. If two variables X_i and X_j are dependent with a specified level of significance α, they are joined by an edge. α is a parameter of the algorithm. The algorithm weights each edge $i \sim j$ in the independence graph with a value $w(i, j)$ stressing the pairwise interaction between the variables. We use the value of the Chi-square test to set $w(i, j)$.

If the independence graph is very dense, the dimension of the cliques will increase beyond a feasible limit. It is important to impose a limit r to the size of the maximum clique. An alternative solution to this problem is to make the graph sparser in one step previous to the calculation of the cliques. This has been done by allowing a maximum number $r - 1$ of incident edges to each vertex. If one vertex has more than $r - 1$ incident edges, those with the lowest weights are removed. In this way, the size of the maximum clique will always be smaller or equal than r. To find all the maximal cliques of the graphs the Bron and Kerbosch algorithm [3] is used. Junction graphs and junction trees can be constructed using a subset of these cliques [18].

Since in the general case, the partition function Z is not known, we use GS to sample points from $k(\mathbf{x})$. VS, Cy, and In are defined as the parameters of the GS algorithm. VS is the type of visitation scheme, and defines the way in which the variables are selected for update. Random ($VS = 0$), or fixed ($VS = 1$) visitation schemes can be used. Cy is the number of cycles of the GS algorithm. One cycle comprises the update of n variables. In is a parameter that determines the way the initial vector of the GS is constructed. The vector where the GS starts from can be randomly selected ($In = 0$), or sampled from an approximate factorization found using a chordal subgraph of the independence graph ($In = 1$). More details about the GS algorithm can be found in [19].

6.5 Probabilistic Operators

From a given independence graph we will define five different classes of factorizations. We will call these classes *probabilistic operators*. To further specify and control their behavior we will employ parameters r, α, Cy and In. Parameters r and α are general parameters because they impose constraints to the independence graph. These constraints influence the type of factorizations. For instance, if $r = 1$ the graph will be disconnected and the only possible factorization is the univariate, similarly if $r = 2$, the graph will be a set of isolated nodes, paths and cycles. Notice that r represents a constraint on the *maximum clique* of the graph. Manipulating α the density of the graph can be changed, influencing the number and size of the factors. Parameters Cy and In will only affect the Kikuchi approximations. A description of the probabilistic operators follows.

- MK0: A Kikuchi approximation that uses as starting vector for GS a vector sampled from an invalid junction-graph-based factorization.
- MK1: An invalid junction-graph-based factorization.
- MK2: A Kikuchi approximation that uses as starting vector for GS a random vector.
- MK3: An invalid junction-tree-based factorization.
- MK4: A Kikuchi approximation that uses as starting vector for GS a vector sampled a valid junction-tree-based factorization.

7 Experiments

The objectives of our experiments are twofold. The first is to study the influence of the different probabilistic operators in the dynamics of the search and the way they interact. Our analysis will be based on the descriptive measures presented in Section 4. The second goal is to extract from this analysis a number of rules that can be translated to the definition of adaptive EDAs. To evaluate the algorithms, we have selected some difficult instances of the satisfiability (SAT) problem.

7.1 SAT Problem

Let $U = \{u_1, \cdots, u_n\}$ be a set of n Boolean variables. A (partial) truth assignment for U is a (partial) function $T : U \rightarrow \{true, false\}$. Corresponding to each variable u are two literals, u and $\neg u$. A literal u (resp. $\neg u$) is $true$ under T if $T(u) = true$ (resp. $T(u) = false$). We call a set of literals a clause, and a set or sequence (tuple) of clauses a formula. Let ϕ be a formula and C a clause in ϕ. We say that a truth assignment T for U satisfies C if at least one literal $u \in C$ is true under T, and T satisfies ϕ if it satisfies every clause in ϕ. The satisfiability problem is the problem of finding a solution for a formula.

In our representation, variable X_i is associated to the Boolean variable u_i, and $(u_i = true) \Leftrightarrow (x_i = 1)$. As the objective function we use the sum of clauses satisfied by the solution.

As as set of instances, we have used the uniform random-3-SAT, which is a family of SAT problems distributions obtained by randomly generating 3-CNF formulae. The test-set $uf-75$ comprises 1000 instances sampled from the phase transition region of uniform Random-3-SAT. The instances, as well as a detailed explanation about the way they were generated, can be found in the SATLIB benchmark[1]. Each instance in $uf-75$ has 75 variables with 325 clauses.

7.2 Parameters of the Algorithms

In all the experiments, we use truncation selection with parameter $T = 0.15$. The population size was $N = 500$. The best solution in each generation is passed to

[1] http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/Benchmarks/SAT/RND3SAT/descr.html

the new generated population. The maximum number of generations was set to 250. The algorithm stops when the optimum is found or the maximum number of generations is reached. We notice that the maximum number of evaluations is relatively small for reaching the optimum of some of the instances considered. However, our goal was not optimize the parameters of the algorithms but to analyze, for these parameters, the effect of the probabilistic operators. Otherwise noticed, we execute 100 runs of each algorithm.

In the experiments we considered three different scenarios. *Random EDA*, in which at each generation, the probabilistic operators and/or the parameters that will be applied are randomly determined. Otherwise stated, the uniform distribution is used for randomly selecting from the parameter set of possible values. This random scenario is conceived for collecting information about the role of the different probabilistic operators. In the *static EDA* scenario, an EDA with fixed probabilistic operator and parameters is run. These cases are considered as a reference for contrasting results. Finally, in the *adaptive EDA* scenario, an EDA with varying probabilistic operators and/or parameters is run.

7.3 Numerical Experiments

In the first experiment, we consider a random EDA scenario where, at each generation, one of the five probabilistic operators presented in Section 6.5 is randomly selected. Fixed parameters were $\alpha = 0.7$ and $r = 8$. Once the operator has been determined, the parameter Cy is randomly selected. For MK0 and MK4, $Cy \in \{1, \ldots, 6\}$, for MK2, $Cy \in \{10, 20, 30\}$. From 100 runs of the algorithm, information about 88369 generations is collected. For each generation, we have the operator applied and it is possible to compute the response to selection it causes. Figure 2 shows the histogram of the number of times that each of the probabilistic operators causes a positive response to selection (i.e. an increase in the average fitness of the population is achieved). Two main features can be noticed from the graph. First, in terms of $R(t)$, the performance of $MK2$ operator is the worst. On the other hand, the effects operators $MK0$ and $MK4$ are very similar. The same fact holds for operators $MK1$ and $MK3$. This may indicate that, for this experiment, a factor graph does not support more relevant information about the interactions in the graph than that that can be represented by a junction tree.

In Table 2, the change in $R(t)$ is detailed. We compute the average of $R(t)$ for each of the probabilistic operators and each of the corresponding Cy values. The best values of the response to selection are reached for operators MK1 and MK3. For operators MK0 and MK4, $R(t)$ decreases with higher values of Cy. For MK2, the decrease in the response of selection is slightly slowed down when Cy is increased. In general, the average results seem to indicate that valid factorizations guarantee higher values of $R(t)$. However, average results can be deceptive. Therefore, we have computed a classification tree to determine the best suited probabilistic operators according to the variance of the population. To compute the classification tree, the treefit procedure implemented in the Matlab software has been employed.

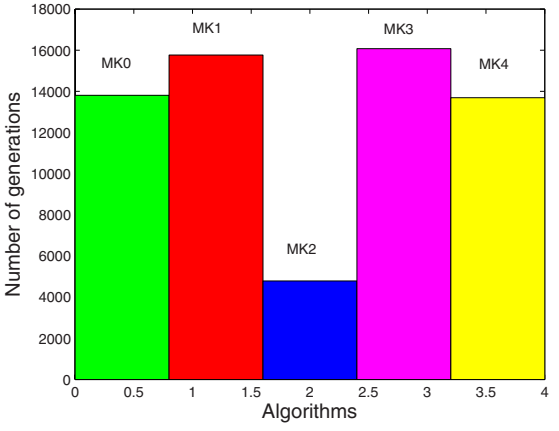


Fig. 2. Average response to selection for different probabilistic operators

Table 2. Average response to selection for different probabilistic models of an EDA with multiple models and sampling algorithms

Cy/EDAs	MK0		MK1		MK2		MK3		MK4	
	mean	σ^2	*mean*	σ^2	*mean*	σ^2	*mean*	σ^2	*mean*	σ^2
1	3.51	9.26	3.68	8.78			3.68	9.00	3.47	9.60
2	3.27	11.62	3.61	8.95			3.65	9.20	3.28	12.67
3	2.75	15.62	3.64	8.79			3.70	9.24	2.91	14.95
4	2.53	18.74	3.75	8.96			3.78	9.39	2.58	18.53
5	2.12	22.50	3.69	8.79			3.78	9.06	2.29	21.10
6	1.92	25.62	3.77	9.26			3.66	9.10	2.06	23.97
10					−12.72	297.0				
20					−12.64	327.2				
30					−12.23	328.6				
all	2.69	17.50	3.69	8.92	−12.53	317.69	3.72	9.18	2.92	15.53

We have taken as predictor variables, the (discretized) variance and a variable Rs, such that $Rs = 0$ if $R(t) < 0$, and $Rs = 1$ if $R(t) > 0$. The categorical dependent class is the type of probabilistic operator, taking into account the value of Cy. Since only three different values where considered for the operator MK2, we have grouped the values for MK0 and MK4 in three groups ($Cy \leq 2$, $Cy \in \{3, 4\}$, and $Cy \geq 5$). Similarly, occurrences of MK1 and MK3 have been equally divided in three groups, but in this case the membership to the group has no implications for the classification.

Figures 3 and 4 show the computed classification tree. It can be observed that most, although not all, of the choices of probabilistic operators that cause a negative value of $R(t)$ (Figure 3) correspond to the MK2 operators. Conversely,

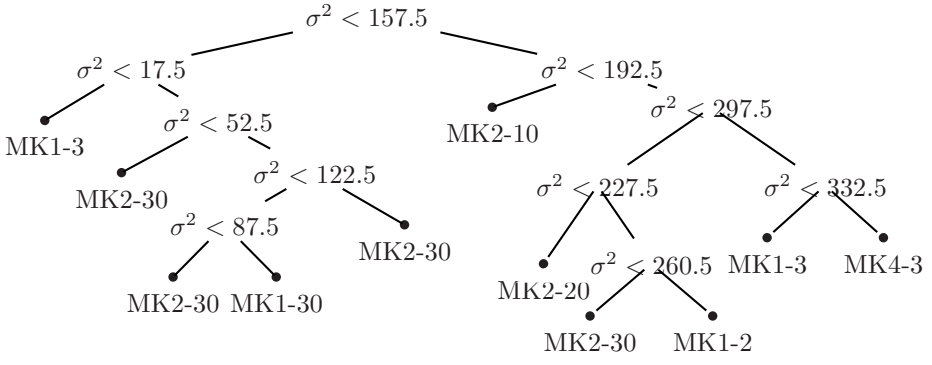


Fig. 3. Classification tree showing the relationship between the variance and the probabilistic operators when the response to selection is negative ($Rs < 0.5$)

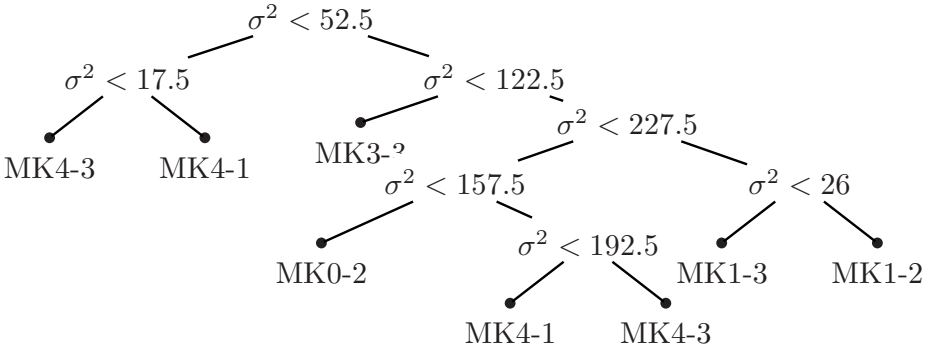


Fig. 4. Classification tree showing the relationship between the variance and the probabilistic operators when the response to selection is positive or zero ($Rs > 0.5$)

MK2 does not appear associated to any value of the variance in the main right branch of the tree (Figure 4).

We have also analyzed the effects that the different probabilistic operators have in the variance of the algorithm. Figure 5 shows the relationship between the response of selection and the variance for MK1 and MK2 operators. This figure reveals an important result. Even if the average value of $R(t)$ is negative for MK2, this operator has a higher variance and may obtain an improvement in the fitness average superior to operator MK1. In simpler terms, operator MK1 regularly improves the solutions but this improvement is constrained. On the other hand, operator MK2 seldom improves solutions but when it does it, the improvement can be important. Additionally, the improvement achieved by operator MK1 is achieved at the cost of an important loss of variance. This is not the case for operator MK2, when the response to selection is improved, also the variance of the generated solutions is increased.

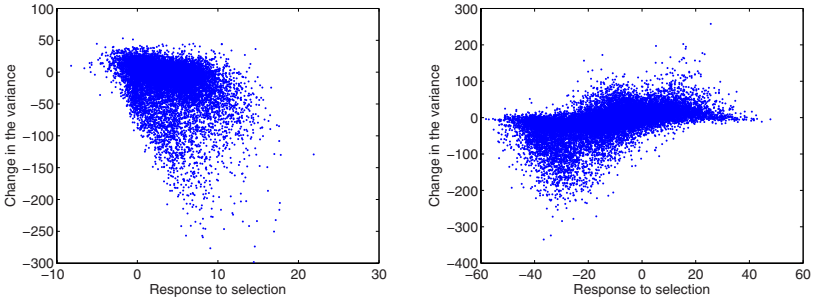


Fig. 5. Relationship between the response to selection and the variance for probabilistic operators MK1 (left) and MK2 (right)

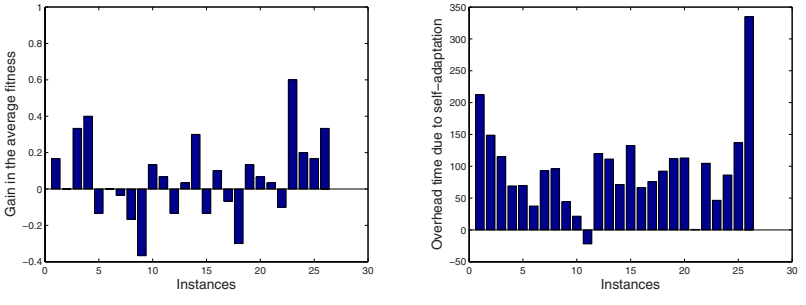


Fig. 6. Gain in the average fitness (left) and overhead time (right) due to the self-adaptation process

We have investigated in our experiments the influence of parameters α and r (data not shown) and extracted decision rules using classification trees. We have evaluated EDAs that incorporate these rules but, for the instances considered, these results are not statistically significant. It turned out, that, at least for the instances of the SAT problem, adaptation based on a combination of the exploratory effect of the MK2 operators with the rest of the operators gives the best results. The resulting algorithm alternates the application of the operators pursuing the goal of balancing exploration and exploitation.

Three different criteria are used to identify a loss of diversity in the population and change the type of probabilistic operator applied. These criteria are: the fitness variance of the selected population is zero, the number of different individuals in the selected set is below half the size of the selected set, and if two consecutive generations have equal average fitness of the selected population.

When one of these criteria is fulfilled, operator MK2 is applied with a randomly selected value of $Cy \in \{1, \ldots, 5\}$. In Table 3, the results for different static and adaptive EDAs and the four instances considered in our experiments are presented. In the table, S is the number of times the optimum has been

Table 3. Success rate and average number for different variants of static and adaptive EDAs

EDAs	scheme	α	r	steps	uf001		uf002		uf003		uf004	
					S	$\bar{f}$	S	$\bar{f}$	S	$\bar{f}$	S	$\bar{f}$
	Random	0.70	8	–	39	324.09	40	324.26	0	322.63	1	322.73
MK1	static	0.92	8	–	43	324.11	46	324.29	0	323.03	0	322.09
MK2	static	0.92	8	2	39	324.06	52	324.49	0	323.11	0	322.74
	adaptive	0.92	8		67	324.66	87	324.87	0	323.80	0	323.64
	adaptive	0.92	6		47	324.40	57	324.53	0	323.39	0	323.16
	adaptive	0.92	4		62	324.59	67	324.66	1	323.40	2	323.23
	adaptive	0.92	2		62	324.59	59	324.58	0	323.36	2	323.35
	adaptive	0.70	1		35	323.90	67	324.63	0	323.27	0	323.00
	adaptive	0.70	3		50	324.43	68	324.68	1	323.34	2	323.20
	adaptive	0.70	6		45	324.19	50	324.46	0	323.36	0	322.99

found and $\bar{f}$ the average fitness of the best found solution. Notice, that for instances $uf003$ and $uf004$ the optimum is very difficult to find. In these cases, we take $\bar{f}$ to evaluate the performance of the algorithms. The random EDA is the algorithm for which previous results have been presented in this section. The adaptive EDAs ($r = 8$) clearly outperforms the other algorithms. The analysis of the table also reveals that factors α and r can play an important role in the performance of the algorithms. Finding schedules for adaptively changing these values during the search should produce more efficient algorithms.

Additional experiments have been conducted for instances from $uf005$ to $uf030$ of the uf-75 benchmark. For these problems, we have compared the performance of the MK1 operator and the adaptive EDA with $\alpha = 0.92$ and $r = 8$ parameters. For each problem, 30 experiments has been conducted from which the average fitness of the best solution found and the overhead time due to the adaptation process have been computed. The results are shown in Figure 6. The adaptive EDA improves the results in 15 of the 26 instances. However, in 9 of the instances worse results are achieved. In 25 of the 26 instances there is a cost due to the adaptation process. Although, the application of the adaptive schedule does not always guarantee an improvement of the results, the improvement achieved can be very important for some of the instances, justifying the additional time spent for the adaptation.

8 Conclusions

In this chapter, we have proposed a general framework for the analysis and design of adaptive EDAs. We have analyzed the main differences between GAs and EDAs regarding the ways adaptation can be incorporated to the algorithms. We have focused on feasible ways of adaptively combining different types probabilistic models in EDAs. Using probabilistic operators based on factor graph based factorizations and Kikuchi approximations we have introduced an

adaptive schedule and evaluated its performance in the optimization of different SAT instances. Our preliminary results show that adaptive EDAs can outperform static EDAs.

The design of flexible, adaptive EDAs, is a difficult challenge that in order to be overcome may require the combination of results from different fields (e.g. data mining, machine learning, automatic control, etc.). However, the benefits to be obtained from this type of algorithms justify the efforts on this trend. We consider the work presented in this chapter as an initial step in this direction.

Acknowledgements

The authors thank the reviewers for useful comments on the paper. This work was supported by the SAIOTEK-Autoinmune (II) 2006 and Etortek research projects from the Basque Government. It has been also supported by the Spanish Ministerio de Ciencia y Tecnología under grant TIN 2005-03824.

References

1. Abbeel, P., Koller, D., Ng, A.Y.: Learning factor graphs in polynomial time and sample complexity. Journal of Machine Learning Research 7, 1743–1788 (2006)
2. Bosman, P.A., Grahl, J.: Matching inductive search bias and problem structure in continuous estimation of distribution algorithms. European Journal of Operational Research (to appear, 2007)
3. Bron, C., Kerbosch, J.: Algorithm 457—finding all cliques of an undirected graph. Communications of the ACM 16(6), 575–577 (1973)
4. Etxeberria, R., Larrañaga, P.: Global optimization using Bayesian networks. In: Ochoa, A., Soto, M.R., Santana, R. (eds.) Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), Havana, Cuba, pp. 151–173 (1999)
5. Grahl, J., Bosman, P.A., Rothlauf, F.: The correlation-triggered adaptive variance scaling idea. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. GECCO 2006, pp. 397–404. ACM Press, New York (2006)
6. Herrera, F., Lozano, M.: Adaptive genetic algorithms based on fuzzy techniques. In: Proceedings of Information Processing and Management of Uncertainty Conference. IPMU 1996, pp. 775–780 (1996)
7. Höns, R., Santana, R., Larrañaga, P., Lozano, J.A.: Optimization by max-propagation using Kikuchi approximations, (submitted for publication, 2007)
8. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
9. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Boston (2002)
10. Mahnig, T., Mühlenbein, H.: Comparing the adaptive Boltzmann selection schedule SDS to truncation selection. In: Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS 2001), Havana, Cuba, March 2001, pp. 121–128 (2001)
11. Mühlenbein, H., Höns, R.: The estimation of distributions and the minimum relative entropy principle. Evolutionary Computation 13(1), 1–27 (2005)
12. Mühlenbein, H., Mahnig, T., Ochoa, A.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics 5(2), 213–247 (1999)

13. 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)
14. Mühlenbein, H., Schlierkamp-Voosen, D.: The science of breeding and its application to the breeder genetic algorithm (BGA). Evolutionary Computation 1(4), 335–360 (1993)
15. Ochoa, A., Soto, M.R., Santana, R., Madera, J.C., Jorge, N.: The Factorized Distribution Algorithm and the junction tree: A learning perspective. In: Ochoa, A., Soto, M.R., Santana, R. (eds.) Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), Havana, Cuba, March 1999, pp. 368–377 (1999)
16. Pettinger, J.E., Everson, R.M.: Controlling genetic algorithms with reinforcement learning. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002, p. 692. Morgan Kaufmann Publishers Inc., San Francisco (2002)
17. Santana, R.: An analysis of the performance of the mixture of trees factorized distribution algorithm when priors and adaptive learning are used. Technical Report ICIMAF 2002-180, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (March 2002)
18. Santana, R.: A Markov network based factorized distribution algorithm for optimization. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 337–348. Springer, Heidelberg (2003)
19. Santana, R.: Estimation of distribution algorithms with Kikuchi approximations. Evolutionary Computation 13(1), 67–97 (2005)
20. Santana, R., Larrañaga, P., Lozano, J.A.: Mixtures of Kikuchi approximations. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 365–376. Springer, Heidelberg (2006)
21. Santana, R., Ochoa, A., Soto, M.R.: The mixture of trees factorized distribution algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2001, pp. 543–550. Morgan Kaufmann Publishers, San Francisco (2001)
22. Schaffer, J.D., Eshelman, L.J.: On crossover as an evolutionarily viable strategy. In: Belew, R.K., Booker, L.B. (eds.) Proceedings of the 4th International Conference on Genetic Algorithms, pp. 61–68. Morgan Kaufmann, San Francisco (1991)
23. Sebag, M., Schoenauer, M.: Controlling crossover through inductive learning. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) Parallel Problem Solving from Nature – PPSN III, pp. 209–218. Springer, Berlin (1994)
24. Smith, J.E., Fogarty, T.C.: Operator and parameter adaptation in genetic algorithms. Soft Computing - A Fusion of Foundations, Methodologies and Applications 2, 81–87 (1997)
25. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory 51(7), 2282–2312 (2005)
26. Zhou, A., Zhang, Q., Jin, Y., Sendhoff, B.: Adaptive modelling strategy for continuous multiobjective optimization. In: Proceedings of the 2007 Congress on Evolutionary Computation CEC 2007. IEEE Press, Singapore (2007)

Initialization and Displacement of the Particles in TRIBES, a Parameter-Free Particle Swarm Optimization Algorithm

Yann Cooren, Maurice Clerc, and Patrick Siarry

Laboratoire Images, Signaux et Systèmes Intelligents, LiSSi, E.A. 3956 Université de Paris XII, 61 avenue du Général de Gaulle, 94010 Créteil, France
{cooren,siarry}@univ-paris12.fr, maurice.clerc@writeme.com

Summary. This chapter presents two ways of improvement for TRIBES, a parameter-free Particle Swarm Optimization (PSO) algorithm. PSO requires the tuning of a set of parameters, and the performance of the algorithm is strongly linked to the values given to the parameter set. However, finding the optimal set of parameters is a very hard and time consuming problem. So, Clerc worked out TRIBES, a totally adaptive algorithm that avoids parameter fitting. Experimental results are encouraging but are still worse than many algorithms. The purpose of this chapter is to demonstrate how TRIBES can be improved by choosing a new way of initialization of the particles and by hybridizing it with an Estimation of Distribution Algorithm (EDA). These two improvements aim at allowing the algorithm to explore as widely as possible the search space and avoid a premature convergence in a local optimum. Obtained results show that, compared to other algorithms, the proposed algorithm gives results either equal or better.

Keywords: Particle swarm optimization, estimation of distribution algorithm, continuous optimization.

1 Introduction

Particle Swarm Optimization (PSO) is a population-based optimization technique proposed by Kennedy and Eberhart in 1995 [6]. Like ant colony algorithms or genetic algorithms, PSO is a biologically-inspired metaheuristic. The method is inspired from the social behavior of animals evolving in swarms, like birds or fishes. The principle is to use collaboration among a population of simple search agents to find the optimum in a function space. More precisely, a simple agent has basically the knowledge of the characteristics of its surroundings but, by communicating with other particles of the swarm, it also has a global knowledge of the search space, as it can be seen in a fish school which tries to find something to eat. PSO is also a particular case in the metaheuristic field because PSO was directly designed for solving continuous problems. This point has its importance because most of the applications deal with continuous problems. A state of the art of PSO and all the concepts which are linked to it is available in [2].

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 199–219, 2008.
springerlink.com

Like other metaheuristics, PSO shows the drawback of comprising many parameters which have to be defined. The difficulty is that the performances of the algorithm are strongly linked to the values given to these parameters. Such a remark implies that it is difficult and time consuming to find the optimal combination of parameter values. Moreover, in real problems, the parameters are often correlated, which makes the choice of parameters harder. So, researches are led to reduce the number of "free" parameters. The aim is to design algorithms which are as efficient as classical algorithms, but with a lower number of parameters. Such algorithms can be called "adaptive algorithms", because information gradually collected during the optimization process are used to compute the values of the parameters. The algorithm is "adaptive" in the way that its behavior, characterized by the values given to the parameters, is evolving all along the process. Several adaptive methods already exist for PSO [15,16,18]. All these algorithms are adaptive but not completely, i.e. there are still parameters to define, so the problem is admittedly easier, but still existing. The ideal would be to design a parameter-free algorithm. A parameter-free algorithm acts as a "black-box" and the user has just to define his problem and to run the process; no parameter has to be defined. Such an algorithm exists among genetic algorithms [10]. Clerc has created a parameter-free algorithm for PSO called TRIBES [3,4]. In this chapter, we will describe the rules of adaptation which permit to avoid the definition of parameters in TRIBES and two ways of improvement will be explored.

Section 2 is dedicated to a brief presentation of the basic PSO algorithm. TRIBES is described in Section 3. In Section 4, we propose a new method to initialize TRIBES. A discussion of the strategies of displacement is presented in Section 5. Some numerical results are shown in Section 6. Finally, we conclude in Section 7.

2 Basic Particle Swarm Optimization

PSO is easy to be coded and implemented. In addition, its simplicity implies that the algorithm is inexpensive in terms of memory requirement and CPU time [4]. All these characteristics have made the popularity of PSO in the field of metaheuristics.

PSO starts with a random initialization of a swarm of particles in the search space. Each particle is modelled by its position in the search space and its velocity. At each time step, all particles adjust their positions and velocities, thus their trajectories, according to their best locations and the location of the best particle of the swarm, in the global version of the algorithm, or of their neighbors, in the local version. Here appears the social behavior of the particles. Indeed, each individual is influenced not only by its own experience but also by the experience of other particles.

In a D-dimensional search space, the position and the velocity of the ith particle can be represented as $X_i = [x_{i,1}, \cdots, x_{i,D}]$ and $V_i = [v_{i,1}, \cdots, v_{i,D}]$ respectively. Each particle has its own best location $p_i = [p_{i,1}, \cdots, p_{i,D}]$, which corresponds to the best location reached by the ith particle at time t. The

global best location is named $g = [g_i, \cdots, g_D]$, which represents the best location reached by the entire swarm. From time t to time t+1, each velocity is updated using the following equation:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_1 (p_{i,j}(t) - v_{i,j}(t)) + c_2 r_2 (g_j(t) - v_{i,j}(t)) \quad (1)$$

where w is a constant called *inertia factor*, c_1 and c_2 are constants called *acceleration coefficients*, r_1 and r_2 are two independent random numbers uniformly distributed in [0,1]. Generally, the value of each component in V_i can be clamped in a range $[-V_{\max}, V_{\max}]$ to control excessive roaming of the particles outside the search space. If the computed velocity leads one particle out of the search space, two methods can be used:

- the particle goes out of the search space but its fitness is not computed.
- the particle is brought back in the search space either on the nearest bound or by applying a multiplicative coefficient chosen in]-1,0[.

The computation of the position at time $t+1$ is derived from Eq.(1) using:

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \quad (2)$$

The inertia weight w controls the impact of the previous velocity on the current one, so it ensures the diversity of the swarm, which is the main means to avoid the stagnation of particles at local optima. In the same way, c_1 controls the attitude of the particle of searching around its best location and c_2 controls the influence of the swarm on the particle's behavior. To summarize, we can say that w controls the diversification feature of the algorithm and c_1 and c_2 the intensification feature of the algorithm. In [5], Clerc and al show that the convergence of PSO may be insured by the use of a constriction factor. Using the constriction factor emancipates us to define $V_{\max}$. In this case, Eq (1) becomes:

$$v_{i,j}(t+1) = K\left(v_{i,j}(t) + \phi_1 r_1 (p_{i,j}(t) - v_{i,j}(t)) + \phi_2 r_2 (g_j(t) - v_{i,j}(t))\right) \quad (3)$$

with:

$$K = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}, \ \phi = \phi_1 + \phi_2, \ \phi > 4 \quad (4)$$

The convergence characteristic of the system can be controlled by ϕ. Namely, Clerc and al. [5] found that the system behavior can be controlled so that the system behavior has the following rules:

- the system does not diverge in a real value region and finally can converge,
- the system can search different regions efficiently by avoiding premature convergence.

Unlike other evolutionary computation methods, constricted PSO ensures the convergence of the search procedure based on the mathematical theory. The standard PSO procedure can be summarized in the algorithm in Figure 1.

Generally, the stopping criterion is either a predefined acceptable error or a maximum "reasonable" number of evaluations of the objective function.

Initialize a population of particles with random positions and velocities.
Evaluate the objective function for each particle and compute g.
For each individual, p_i is **initialized** at X_i.
Do
  Update the velocities and the positions of the particles.
  Evaluate the objective function for each individual.
  Compute the new p_i and g.
While the stopping criterion is not met

Fig. 1. Standard PSO procedure

3 TRIBES, a Parameter-Free PSO Algorithm

Like many other metaheuristics, PSO shows the drawback of having too many parameters which must be set by the user. According to the values given to these parameters the algorithm is more or less efficient. A first approach to adaptive PSO was proposed by Ye [16], whose method looks for inactive particles and replaces them by new particles, more able to explore new areas of the search space. Zhang et al. [18] proposed to modify swarm's size, constriction factor or neighborhood size through the use of an improvement threshold. Yasuda [15] worked out an algorithm in which parameters are defined according to the velocity information of the swarm. But the first parameter-free algorithm, called TRIBES, was proposed by Clerc [3]. TRIBES is an adaptive algorithm whose parameters change according to the swarm behavior. In TRIBES, the user only has to define the objective function and the stopping criterion. The method incorporates rules defining how the structure of the swarm must be modified and also how a given particle must behave, according to the information gradually collected during the optimization process.

3.1 Swarm, Tribes and Communication

PSO is based on the social behavior of animals evolving in swarms. Each individual of the swarm knows the direction of displacement and the velocity of its neighbors in the swarm and uses this information to decide its own direction of displacement and its own velocity. Considering that the swarm is an interconnected network, information collected by one of the individuals is propagated in the entire swarm. So, all the individuals modify their behavior according to the new interesting information. This implies a "global" behavior of the swarm, which allows the swarm to find regions of interest in the search space. These considerations form the framework of Standard PSO.

However, it can also be observed in real life that a swarm can be divided in "tribes" of individuals. Here, the behavior of the swarm is different from the one explained before. Each tribe acts as an independent swarm, i.e. each tribe has its own "global behavior" and explores a particular region of the search space. In addition to that, all the tribes exchange information about regions they are exploring.

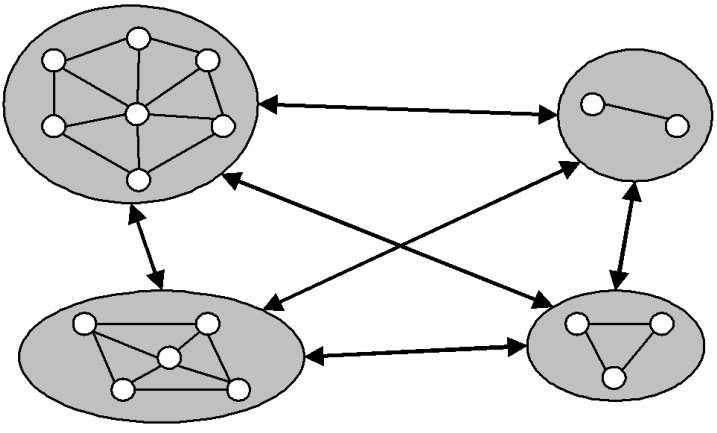


Fig. 2. Intra-tribe and inter-tribes communications

So, the swarm is an interconnected network of tribes, which are themselves interconnected networks of individuals. This implies two different types of communication: intra-tribe communication and inter-tribes communication. These considerations form the framework of TRIBES. In Figure 2, an example of a swarm of 17 particles (white spots) divided in 4 tribes is shown. Arrows symbolize inter-tribes communications and lines symbolize intra-tribe communications.

In TRIBES, the swarm is structured in different tribes of variable size. The aim is to simultaneously explore several promising areas, generally local optima, and to exchange results between all the tribes in view of finding the global optimum.

Each tribe is composed of a variable number of particles. Relations between particles in a tribe are similar to those defined in basic PSO. More precisely, each particle of the tribe stores the best location it has met and knows the best (and the worst) particle of the tribe, i.e. the particle which has met the best (and the worst) location in the search space. This is intra-tribe communication.

Even if each tribe is able to find a local optimum, a global decision must be taken to decide which of these optima is the best one. So, each tribe is linked to all the others to inform them on the best location found by its best particle. This is inter-tribes communication.

TRIBES is an adaptive algorithm, so the swarm must be generated and modified automatically, by means of creation, evolution, and removal of the tribes.

3.2 Structural Adaptations

Setting rules to modify the swarm's structure implies the definition of a quality qualifier for each particle and likewise for the tribes. In the case of particles, it is known that each particle has a current position and a best position. So, a particle is said to be a *good* particle if it has just improved its best performance, *neutral* if not. In addition to this qualitative (because not relative to values,

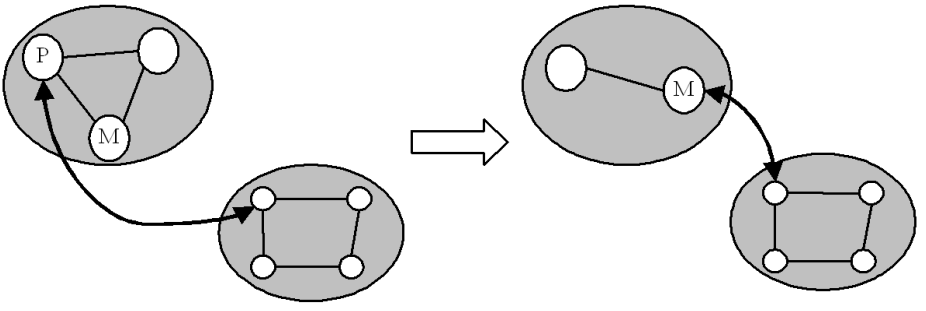


Fig. 3. Removal of a particle from a multiparticle tribe

but to improvement) qualification, the *best* and the *worst* particles are defined within the tribe framework.

In addition, good and bad statuses are also defined for the tribes. These statuses are related to the amount of good particles inside the tribes. It is postulated that: "The larger the number of good particles in the tribe, the better the tribe itself". In practice, a random integer number p between 0 and the swarm's size is generated according to a uniform distribution. Then, if the number of good particles in the tribe is strictly larger than p, the tribe is said *good*, if not, the tribe is said *bad*.

These qualifiers allow us to define the two following rules.

Removal of a Particle

In most common uses of PSO, the most time consuming part of the algorithm is the objective function evaluation. So, it is interesting to carry out the least number of evaluations of the objective function. Consequently, it will be tried to remove a particle of the swarm as soon as possible, on condition that the removal does not affect the final result. That is why a removal of particle must occur in a good tribe and the removed particle is obviously the worst. In Figure 3, the particle P is the worst of its tribe and the tribe was declared good. In this case P is removed and the redistribution of its external links (here, only one symmetrical link) is done on M, the best particle of the tribe. The information links that each particle has with itself are not represented, because they do not play any role here.

In the case of a monoparticle tribe, the removal is only made if one of the "informers", i.e. a particle of another tribe by which the inter-tribes communication is made, has a better performance (see Figure 4). This ensures us to keep the better quality of information. In Figure 4, the monoparticle tribe was declared good, thus the single particle P, which is necessarily the worst of the tribe, even if it is at the same time good, should be removed. But it will be removed only if its best external informer M_P is better than itself. The assumption is indeed that the information carried by P is then less valuable than that carried by M_P.

The removal of a particle implies a change in the information network. All the particles linked to the removed particle are redirected to the best particle of

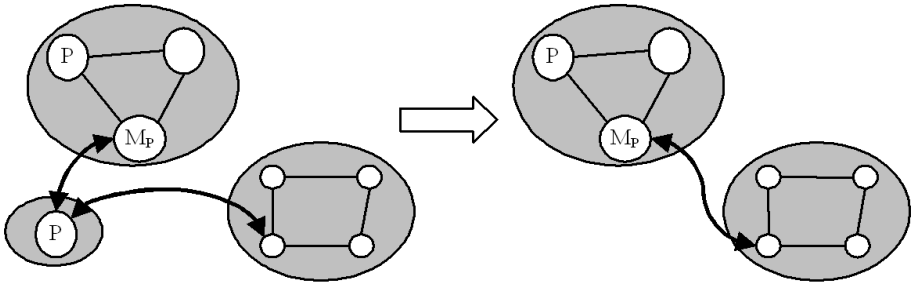


Fig. 4. Removal of a particle from a multiparticle tribe

the removed particle's tribe. In the particular case of a monoparticle tribe, all these links are redirected to the best informer of the removed particle, because removing the particle implies removing the tribe.

Obviously, these structural adaptations must not occur at each iteration, because time must be let for the information propagation. In practice, if NL is information links number at the moment of the last adaptation, the next adaptation will occur after NL/2 iterations.

Generation of a Particle

The process of adding a particle is quite similar to the removal. A bad tribe generates particles which will form a new tribe. The bad tribe will keep the contact with the new tribe and will try to use it to improve its best location.

Three types of particles are generated:

- Free particle: The particle is randomly generated according to a uniform distribution in the whole search space, on a side of the search space or on a vertex of the search space. The idea is to rely on the future course of the particle to cross a promising area.
- Confined particle: If x is the best particle of the generating tribe and i_x the best informer of x, p_x and p_{i_x} are the best locations of x and i_x. The new particle will be generated in the D-sphere of center p_{i_x} and radius $||p_x - p_{i_x}||$. The idea is here to intensify research inside an interesting area.
- Isolated particle: The particle is generated in the biggest "terra incognita", i.e. as far as possible from the existing particles and from the boundaries of the search space. The idea is to explore areas which have not been explored yet and, then, to discover new regions of interest. Figure 5 shows an example of where a "terra incognita" can be located. The * symbolize the particles and the gray regions indicate possible "terra incognita".

Swarm Evolution

At the beginning, the swarm is composed of only one particle which represents a single tribe. If, at the first iteration, this particle does not improve its location, a new one is created, forming a second tribe. At the second iteration, the same process is applied and so on.

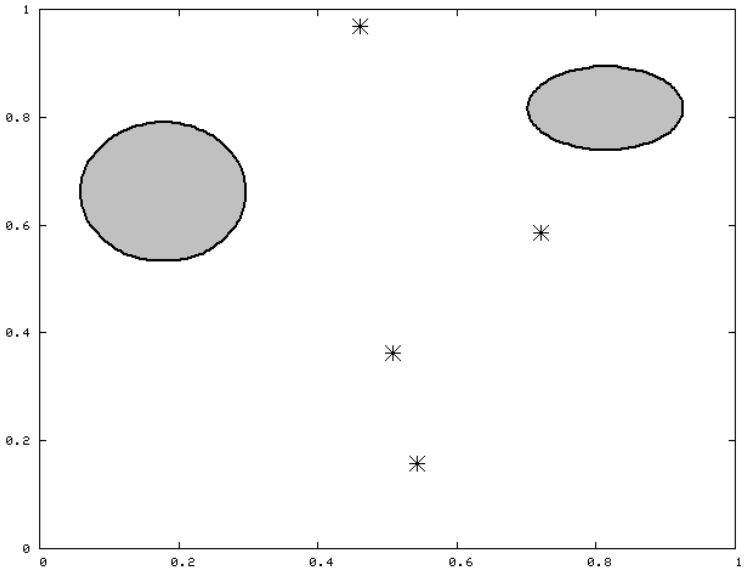


Fig. 5. Location of possible "terra incognita"

The swarm's size will grow up until promising areas are found. The more the swarm grows, the longer the time between two adaptations will be. By this way, the swarm's exploratory capacity will grow up but the adaptations will be more and more spaced in time. Therefore, the swarm has more and more chances to find a good solution between two adaptations.

Contrarily, once a promising area is found, each tribe will gradually remove its worst particle, possibly until it disappears. Ideally, when convergence is confirmed, each tribe will be reduced to a single particle.

3.3 Behavioral Adaptations

In the previous section, the first way of adaptation of the algorithm was described. The second way in view of adapting the swarm to the results found by the particles is to choose the strategy of displacement of each particle according to its recent past. As in the case of the evolution of tribes, it will enable a particle with a good behavior to have an exploration of greater scope, with a special strategy for very good particles, which can be compared to a local search. According to this postulate, the algorithm will choose to call the best displacement's strategy in view of moving the particle to the best possible location it can reach.

There are three possibilities of variation for a particle: deterioration, status quo and improvement, i.e. the current location of the particle is worse, equal or better than its last position. These three statuses are denoted by the following symbols: - for deterioration, = for status quo and + for improvement. The history

of a particle includes the two last variations of its performance. For example, an improvement followed by a deterioration is denoted by (+ -). So, there are nine possibilities of history. The strategy of displacement of a particle will be determined by its couple of variations. The different strategies of displacement will be discussed in Section 5.

To sum up, it can be said that TRIBES is an algorithm which tries to solve one of the main problems of metaheuristics: the fitting of parameters. By adapting the swarm's form and particles' strategies of displacement, TRIBES frees users of defining parameters and acts as a "black box". The particles use their own history and swarm's history to decide how they must move and how the swarm must be organized in view of approaching as efficiently as possible the global optimum. The algorithm in Figure 6 shows a pseudo-code which summarizes TRIBES process. g is the best location reached by the swarm and the p's are the best locations for each particle. NL is the number of information links at the last swarm's adaptation and n is the number of iterations since the last swarm's adaptation.

Initialize a population of particles with random positions and velocities.
Evaluate the objective function for each particle and compute g.
For each individual, p_i is **initialized** at X_i.
Do
  Determination of statuses for each particle.
  Choice of the displacement strategies.
  Update the velocities and the positions of the particles.
  Evaluate the objective function for each individual.
  Compute the new pi and g.
  If $n < NL$
      Determination of tribes qualities
      Swarm's **adaptations**
      Computation of NL
  End if
While the stopping criterion is not met

Fig. 6. TRIBES procedure

4 Initialization of TRIBES

The efficiency of every PSO-inspired algorithm is linked to the initialization of the particles. In [1], it is proved that, in basic PSO, the position of a particle at the time step t can be decomposed in two vectors, one which only depends on the initial configuration and one which does not depend on the initial configuration. The trajectory of a given particle is then linearly dependent from its initial position and its initial velocity. So, it clearly appears here that a good choice of initial positions and velocities can lead to better results. The ideal case would be to assess the initial points in the way that the search space is explored as widely as possible by the trajectories of the particles.

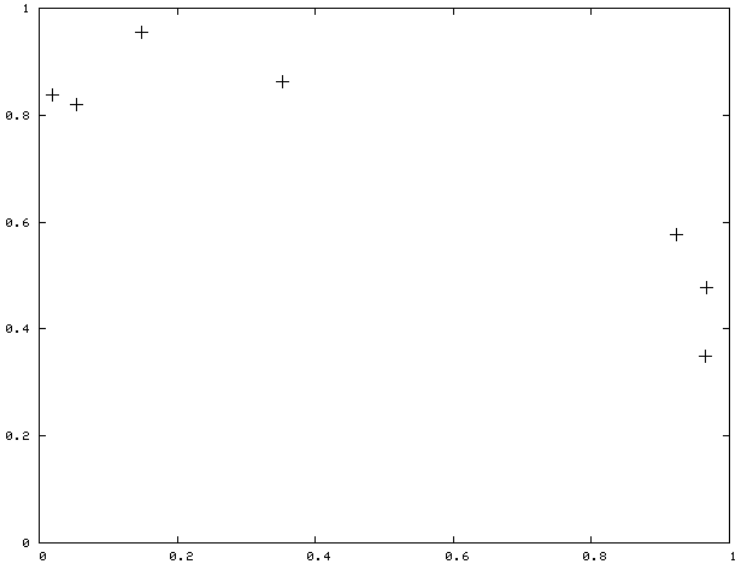


Fig. 7. Random initialization

Commonly, particles are randomly initialized in the search space. But, such an initialization can lead to a bad diversity of the particles. On Figure 7, a random initialization of seven particles in dimension D=2 is shown. It can be seen that the particles are confined in the lower-right part of the search space and in the upper-left part. In this way, the particles would first explore the upper-left and the lower-right corners of the search space and, then, there is a possibility that they would be trapped in a local optimum situated in this region without having explored the other regions of the search space.

In view of avoiding this problem, a new way of initialization is proposed. The idea is to fill as widely as possible the search space. To summarize, the particles will be initialized so that each particle will be as far as possible from the others and as far as possible from the boundaries of the search space.

In dimension D, TRIBES will be initialized with $D+1$ particles. The initialization is made using a standard particle swarm optimization using the objective function of Eq.(5):

$$f = \sum_{i=1}^{D+1} \sum_{j \neq i} \frac{1}{d_{ij}} + \sum_{i=1}^{D+1} \frac{1}{\min_{d \in [1..D]}(d(x_i, bound_d))} \tag{5}$$

where d_{ij} is the distance between particle i and particle j and $d(x_i, bound_d)$ is the distance between particle i and the boundary of the dth dimension.

Figure 8 shows that this new way of initialization leads to a better diversity of the particles in the search space. The particles fill well the search space and no region will be preferred to others during the exploration process.

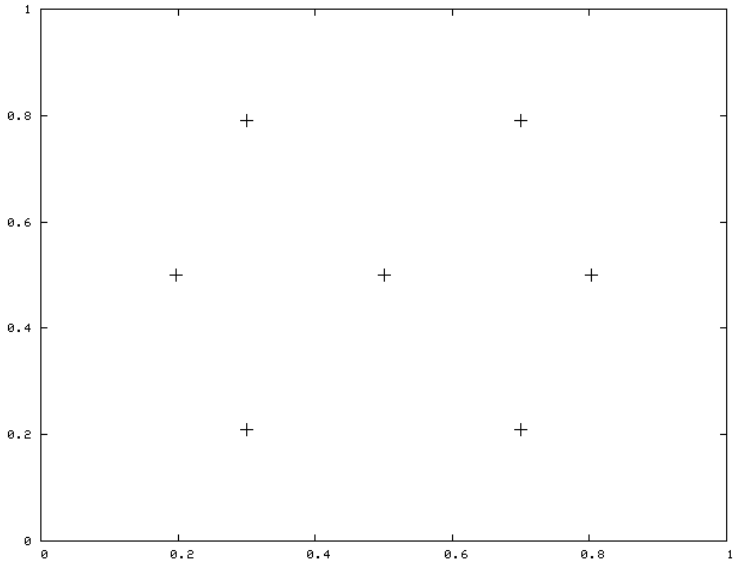


Fig. 8. Regular initialization

5 Strategies of Displacement

It was seen in Section 4 that the way of initializing the particles has its importance in view of making PSO as effective as possible. Once the particles are initialized, it must be decided how they will move. In basic PSO, the strategy of displacement is the same for each particle. Eqs.(1,2) model this strategy.

In TRIBES, the strategy of displacement is different for each particle and can be modified at each time step. For a given particle, the choice of its strategy of displacement is made according to its recent history. In this section, the original strategies defined by Clerc in [4] are exposed and a new strategy is defined.

5.1 Basic Strategies of TRIBES

It was said in Section 3.3 that the strategy of displacement of a given particle depends on its two last variations. So, there are nine possibilities of history. Clerc [4] gathered them in three groups. So, only three strategies are needed. The three used strategies are the following:

- Pivot strategy: This method is inspired from [11]. Let us denote by p the best location of the particle, g the best position of the informers of the particle and f the objective function. The movement is done as follows:

$$X = c_1 U(H_p) + c_2 U(H_g) \tag{6}$$

with $c_1 = f(p)/(f(p) + f(g))$, $c_2 = f(g)/(f(p) + f(g))$, $U(H_p)$ a point uniformly chosen in the hyper-sphere of center p and radius $||p - g||$ and $U(H_g)$ a point uniformly chosen in the hyper-sphere of center g and radius $||p - g||$.

- Disturbed pivot strategy: Here we have

$$X = c_1 U(H_p) + c_2 U(H_g) \tag{7}$$

$$b = N(0, \frac{f(p) - f(g)}{f(p) + f(g)} \tag{8}$$

$$X_j = (1 + b)X_j \tag{9}$$

- Local strategy by independent gaussians: If g is the best particle of the swarm, we have

$$X_j = g_j + N(g_j - X_j, ||g_j - X_j||) \tag{10}$$

where $N(g_j - X_j, ||g_j - X_j||)$ is a point randomly chosen with a monodimensional gaussian distribution of mean $g_j - X_j$ and standard deviation $||gj - Xj||$.

So, (= +) and (+ +) represent the best particles and it will be chosen for them a local strategy by "independent gaussians". (+ =) and (- +) represent the neither bad nor good particles and it will be chosen a "disturbed pivot strategy". At last, (- -), (= -), (+ -), (- =) and (= =) represent feeble particles. A "pivot strategy" is chosen for them. If the movement implies one particle to go out the search space, the particle will be brought back to its closest position in the search space. This is particle's confinement.

It can be seen that the strategies of displacement defined in this paragraph are different from the one of Standard PSO, especially because there is no need to define velocity vectors, but the philosophy is almost the same: a particle moves according to its own best performance and according to the best performance of the swarm.

5.2 A New Strategy of Displacement

In the previous paragraph, the three strategies of displacement employed in the first version of TRIBES were defined. Results obtained with the original algorithm can be found in [19]. In view of evaluating the efficiency of the strategies of displacement defined above, a study of the behavior of the particles must be done. No complete theoretical study already exists because the interactions between particles made the problem very hard to solve although a few theoretical results about the dynamics of PSO [5,13,14] are available in the literature. In consequence, only experimentation can inform us about the dynamics of the particles during the process.

Figure 9 shows us examples of "convergence graphs" of TRIBES. Such graph shows the median performance of the total runs, here 25 runs, as a function of the number of evaluations of the objective function (Fes). A semilog scale is used. These graphs are made with Griewank (F7), Rastrigin (F9) and Weierstrass (F11) functions [12] in dimension D=10. It can be seen that TRIBES converges quickly at the beginning of the process and seems to stagnate after. It can

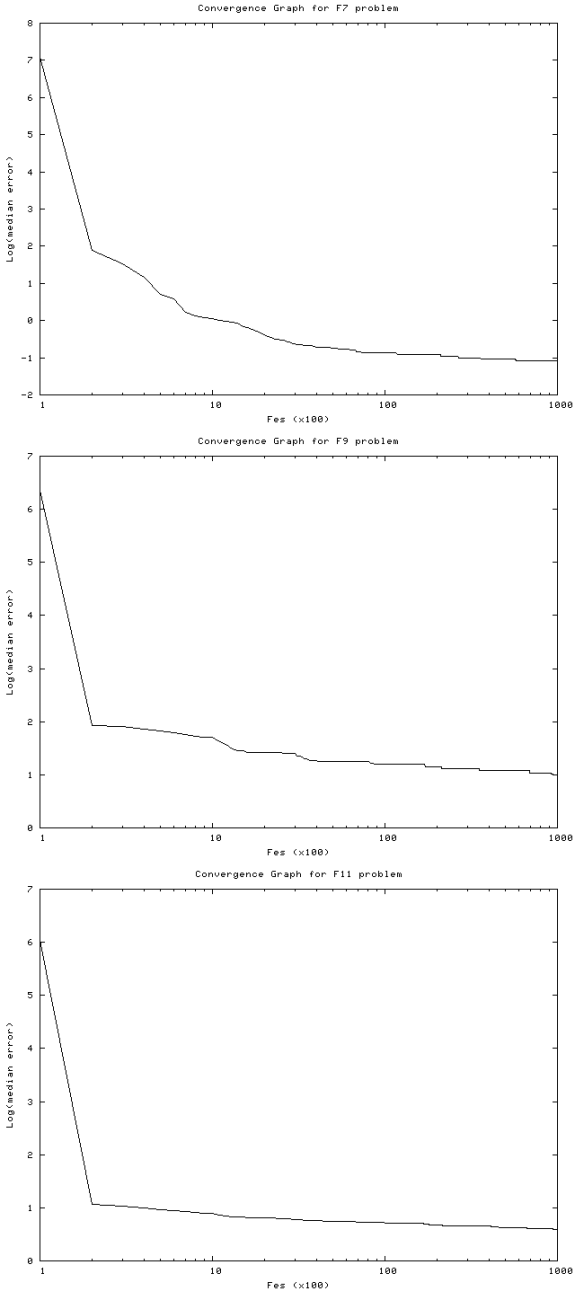


Fig. 9. Convergence graphs for Griewank (top), Rastrigin (middle) and Weierstrass (bottom) functions

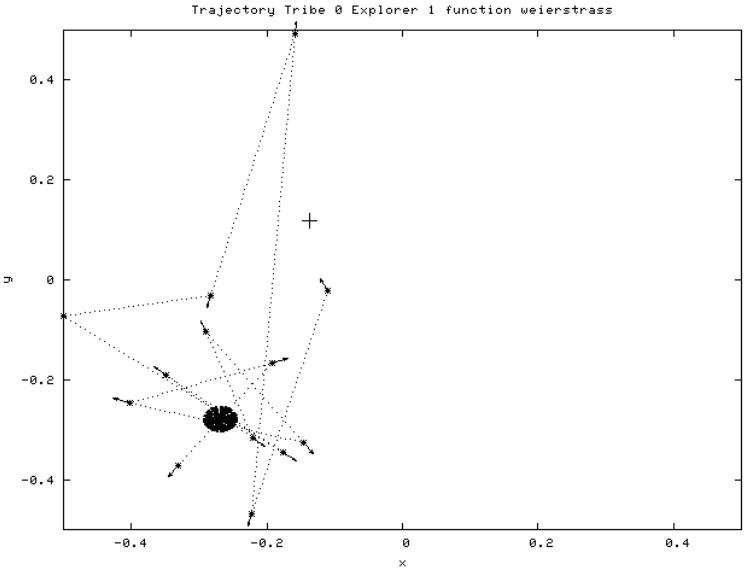


Fig. 10. Trajectory of a particle. The cross indicates the optimum.

Initialize the population X^0
Build the distribution P^0
$X^t = X^0$
$P^t = P^0$
Do
  Select a subgroup X^t_{sub} of X^t
  Build P^{t+1} according to X^t_{sub}
  Sample P^{t+1} to create $X^{t+1}_{offspring}$
  Replace individuals of X^t by individuals of $X^{t+1}_{offspring}$ for creating X^{t+1}
  $t = t + 1$
While the stopping criterion is not met

Fig. 11. Principle of an EDA

be concluded that the particles converge quickly to a local optimum and do not manage to escape from it. This fact is a problem common to every PSO-inspired algorithm. Considering that, in TRIBES, the position of a given particle is randomly chosen using two hyperspheres of centers g and p, the particle is attracted towards these two points. This implies that, if g or p is a local optimum, the particle would be attracted to the "valley" of this optimum, move around it and, so, it would not be possible for the particle to improve its performance. Figure 9 shows an example of this behavior. Figure 9 represents the trajectory of a particle for Weierstrass 2D problem [12]. Arrows are the velocities of the particle at each time step and the cross symbolizes the location of the global

optimum. It is clear that the particle is trapped in a local optimum and moves around it. The observation of the velocity vectors confirms this remark.

Previous remarks imply that TRIBES is defective in matter of diversification, i.e. the search space is not widely explored by the particles. The objective of the new strategy of displacement is to improve the diversification capacity of TRIBES. To reach this goal, an Estimation of Distribution Algorithm (EDA) is used [7]. The principle of EDAs is to select a subgroup of a family of solutions, build a probabilistic model of this subgroup and construct new solutions by randomly sampling the constructed distribution. First, a family X^0 of solutions is randomly generated and its probabilistic model P^0 is constructed. For a given time t, the main loop is composed of four steps. First, a subgroup X^t_{sub} of X^t is chosen using a predefined criterion. Then, according to the element of X^t_{sub}, the probabilistic model P^{t+1} is constructed. At this moment, P^{t+1} is sampled to give the new family of solutions X^{t+1} offspring. Finally, individuals of X^t are replaced by individuals of X^{t+1} offspring. The process is iterated until the predefined stopping criterion is reached. Many different probabilistic models can be used considering or not that the variables are correlated [7]. Algorithm 11 summarizes this process.

Introducing a new strategy of displacement based on EDA can permit us to improve the diversification process of TRIBES. Indeed, the displacement of a given particle will not be driven only by three positions (the current position of the particle, p and g) but by a sub-family of the swarm. So, premature convergence, caused by the stagnation of p and g, is avoided. Moreover, sampling a probabilistic model implies that all the positions of the search space can be reached with a non-zero probability and, then, gives the possibility to a particle to escape from a local optimum in which it was trapped. By this way, the search space can be more widely explored.

The problem is now to choose the appropriate family of particles in view of building the distribution of probability. In a PSO-inspired algorithm like TRIBES, the most obvious choice is to choose the best position of each particle i.e. the p_i. Dimensions are supposed independent, thus a monodimensional probabilistic model is computed for each dimension of the search space. If, at the current time step, the size of the swarm is N and d is the current dimension, the distribution of probability is supposed normal and is modelled by its average and its standard deviation according to Eqs.(11, 12).

$$\mu_d = \frac{1}{N} \sum_{i=1}^{D} p_{i,D} \tag{11}$$

$$\sigma_d^2 = \frac{1}{N-1} \sum_{i=1}^{D} (p_{i,D} - \mu_d)^2 \tag{12}$$

Then, the new coordinate of the particle for the dimension d is randomly chosen according to the normal distribution of average μ_d and standard deviation σ_d.

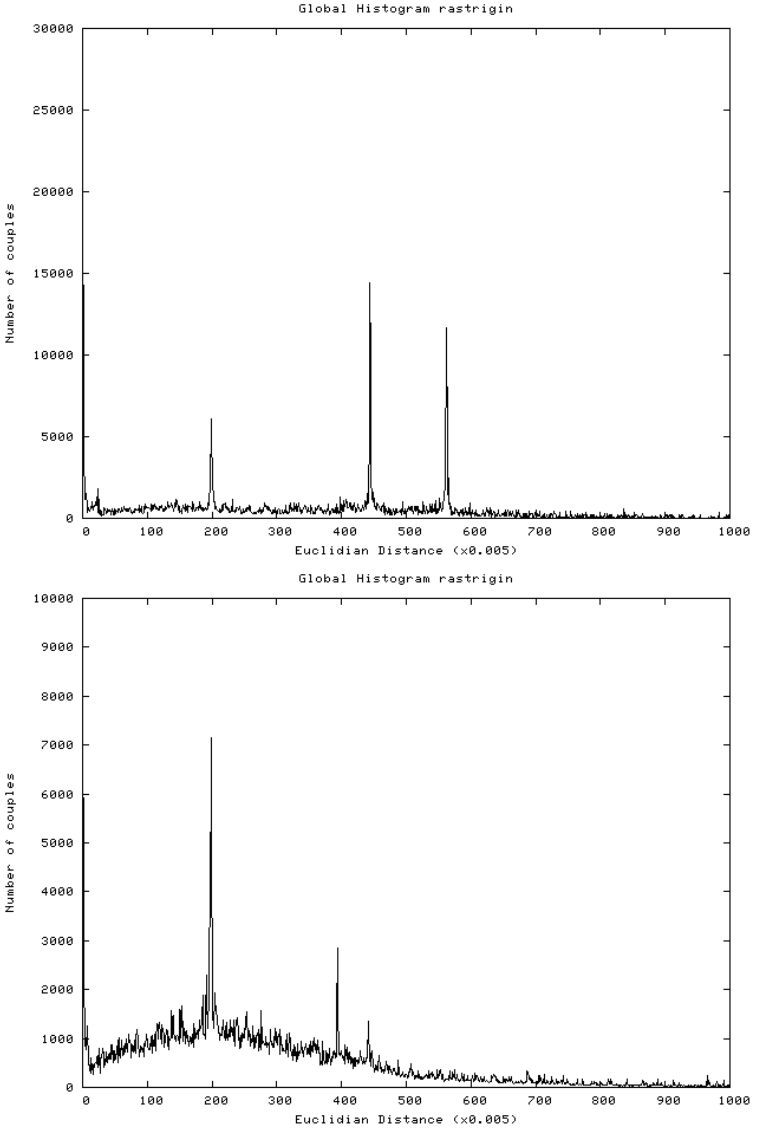


Fig. 12. Histograms of the distance to all reached positions

In real problems, it is current that the variables are correlated. In this case, a joint normal distribution is used instead of D monodimensional normal distributions.

Obviously, this method cannot be employed at any moment for each particle. It had been said above that there are nine possibilities of history for a particle. In

basic TRIBES, histories are gathered in three groups. With the addition of our new strategy of displacement, a new group is created. The use of the new strategy of displacement is dedicated to the worst particles. So, particles with a history (- -), (- =) or (= -) will use the new strategy to compute their displacement.

Figure 12 shows histograms of the distances between all the positions reached by the particles for the Rastrigin 2D problem [12]. It can be seen that, with the utilization of the new strategy of displacement, the positions reached by the particles are less close than in the basic case. This remark implies than the diversification process is better with the utilization of the new strategy. In Figure 12(top), it can be seen that the histogram is mainly composed of three thin peaks. This implies that, at the end of the procedure, the swarm is composed of three tribes which have explored three different regions of the search space. The fact that the peaks are very thin implies that all the particles of a same tribe are concentrated on a local optimum and do not manage to escape from it. Contrarily, in Figure 9b, the histogram is also composed of three peaks but the peaks are, in this case, larger. This implies that the particles have explored more largely the search space and did not stay concentrated around a local optimum. So, it can be concluded that the diversification process of TRIBES is improved.

6 Numerical Results

The aim of this section is to compare Improved TRIBES, implemented with the regular initialization and the new strategy of displacement, with a Standard PSO algorithm [20], a simple continuous EDA [17], a real-coded Memetic Algorithm [8], a real-coded Differential Evolution Algorithm [9] and with Basic TRIBES. Tests are made in dimension D=10. For Standard PSO, the number of neighbors for each particle is 3, the size of the swarm is 20, $w = 1/(2\log 2)$ and $c_1 = c_2 = 1/2 + \log 2$. The process stops if the error is lower or equal to 10^{-6} for functions of Table 1 and 10^{-2} for functions of Table 2 or if the number of evaluations of the objective function exceeds 105. Functions used are extracted from the CEC'2005 benchmark functions set [12]. Table 1 and Table 2 give the mean error of each algorithm for 25 executions and the mean number of evaluations of the objective function, in brackets, respectively for unimodal functions and multimodal functions.

Table 1 and Table 2 show that Improved TRIBES, implemented with the regular initialization and the new strategy of displacement, gives competitive results. Ackley bounds function can be excluded from the comparison because it can be seen in Table 2 that all the algorithms are trapped in the same local optimum. Improved TRIBES solves 50% of the benchmark, the best algorithm being Differential Evolution with 60% of the benchmark solved.

Compared to Standard PSO, it can be seen that Improved TRIBES gives better results on most cases. Indeed, Standard PSO is better only on Schwefel and Schwefel noise, what can be explained by the simplicity of the function and by the fact that Standard PSO starts with more particles than Improved TRIBES. Improved TRIBES is better than TRIBES on 80% of the benchmark.

Table 1. Comparison between several algorithms for unimodal functions

	Sphere	Schwefel	Elliptic	Schwefel noise	Schwefel bounds
Standard PSO	0.00 (3375)	0.00 (8300)	71.28e3 (100000)	0.00 (11562)	18.86 (67610)
Simple EDA	0.00 (28732)	0.00 (28916)	0.00 (32922)	0.00 (32636)	233.50 (100000)
Memetic algorithm	0.00 (11737)	0.00 (36598)	4.77e4 (100000)	0.00 (71563)	0.02 (100000)
Differential Evolution	0.00 (7120)	0.00 (21800)	0.00 (8970)	0.00 (26100)	0.00 (25300)
Basic TRIBES	0.00 (1856)	0.00 (10452)	18.77e3 (100000)	0.00 (18966)	1.74 (60409)
Improved TRIBES	0.00 (1521)	0.00 (12011)	13.11e4 (100000)	0.00 (23783)	9.07e-4 (80832)

Table 2. Comparison between several algorithms for multimodal functions

	Rosen-brock	Grie-wank	Ackley bounds	Rastrigin	Rastrigin rotated	Weier-strass
Standard PSO	1.88 (100000)	0.08 (100000)	20.11 (100000)	4.02 (100000)	10.18 (100000)	4.72 (100000)
Simple EDA	0.00 (43500)	0.53 (10000)	20.33 (100000)	32.28 (100000)	32.28 (100000)	8.27 (100000)
Memetic algorithm	1.48 (100000)	0.19 (100000)	20.19 (100000)	0.43 (100000)	5.63 (100000)	4.55 (100000)
Differential Evolution	0.00 (26100)	0.15 (100000)	20.40 (100000)	0.00 (5110)	36.00 (100000)	4.67 (100000)
Basic TRIBES	0.06 (100000)	0.07 (100000)	20.32 (100000)	8.55 (100000)	12.11 (100000)	5.68 (100000)
Improved TRIBES	0.12 (100000)	0.02 (100000)	20.35 (100000)	0.19 (72584)	8.55 (100000)	3.55 (100000)

Compared to Basic TRIBES, Improved TRIBES is better excepted on Schwefel, Schwefel noise and Rosenbrock functions. On Schwefel and Schwefel noise, it can be deduced that the new strategy of displacement slows a little bit the algorithm mainly because the diversification process is larger. On Rosenbrock function, the result is a little better in favor of Basic TRIBES but the difference is not significant.

Table 1 shows that, on unimodal problems, Improved TRIBES is faster than non-PSO methods except on Elliptic problem. PSO-inspired methods fail totally on this problem whereas Simple EDA and Differential Evolution solve it quite simply. Globally, PSO-inspired methods seem to be more efficient, i.e. quicker, than the others on simple problems easy to solve (ex: Sphere, Schwefel,...).

Table 3. Time comparison between Basic TRIBES and Improved TRIBES for unimodal functions

	Sphere	Schwefel	Elliptic	Schwefel noise	Schwefel bounds
Basic TRIBES	12.72	32.12	21.96	24.18	65.7
Improved TRIBES	26.57	25.28	27.72	34.85	74.12

Table 4. Time comparison between Basic TRIBES and Improved TRIBES for multimodal functions

	Rosen-brock	Grie-wank	Ackley bounds	Rastrigin	Rastrigin rotated	Weier-strass
Basic TRIBES	26.5	24.48	36.6	22.14	32.11	123.55
Improved TRIBES	32.19	32.32	47.2	38.74	46.84	147.66

Table 2 does not permit to make strong conclusions. However, Improved TRIBES appears to be better than PSO-inspired methods and competitive with non-PSO methods.

Table 3 and Table 4 present execution times, in seconds, of 10^5 evaluations of the objective functions for Basic TRIBES and Improved TRIBES. Table 3 and Table 4 show that Improved TRIBES is slower than Basic TRIBES. This fact is easily understandable since the new strategy of displacement needs more computation time than the others. Moreover, the regular initialization process is also slower than a simple random initialization.

7 Conclusions

This chapter has shown that TRIBES uses structural and behavioral rules to avoid the fitting of parameters. Having no parameters implies no lost of time to fit the parameters, which is a very hard problem, but also implies to give more information to the particles, because the process is not "driven" by the values of the parameters.

Two methods are proposed to try to help the particles to explore as widely as possible the search space. The first one is a regular initialization of the particles, which aims at filling as regularly as possible the search space. The second one is a new strategy of displacement based on an estimation of distribution algorithm. This new strategy permits to the particles to have more various displacements, thus avoiding to be trapped into local optima.

Numerical results show that Improved TRIBES is equivalent or better than Standard PSO and Basic TRIBES. Improved TRIBES also appears to be competitive with non-PSO methods. Numerous improvements on TRIBES still are possible. The rules of adaptation are quite simple and use very few information

compared with all the available information. New rules can be defined to try to have the best adaptation of the choices made to the specificity of the problem.

References

1. Campana, E.F., Fasano, G., Peri, D., Pinto, A.: Particle Swarm Optimization: Efficient Globally Convergent Modifications. In: III European Conference on Computational Mechanics, Solids and Coupled Problems in Engineering, Lisbon, Portugal, June 5-8 (2006)
2. Eberhart, R.C., Kennedy, J., Shi, Y.: Swarm Intelligence. In: Evolutionary Computation. Morgan Kaufmann, San Francisco (2001)
3. Clerc, M.: TRIBES - Un exemple d'optimisation par essaim particulaire sans paramtres de contrle. In: OEP 2003, Paris, France (2003)
4. Clerc, M.: Particle Swarm Optimization, International Scientific and Technical Encyclopedia (2006)
5. Clerc, M., Kennedy, J.: The particle swarm: explosion, stability, and convergence in multi-dimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
6. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimisation. In: Proceedings of the IEEE International Conference On Neural Networks, WA, Australia, pp. 1942–1948 (1995)
7. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms, a new tool for evolutionary computation. Kluwer Academic Publishers, Dordrecht (2001)
8. Molina, D., Herrera, F., Lozano, M.: Adaptive Local Search Parameters for Real-Coded Memetic Algorithms. In: Proceedings of the 2005 Conference on Evolutionary Computation, Edinburgh, Scotland, September 2-5, 2005, pp. 888–895 (2005)
9. Rnkknen, J., Kukkonen, S., Price, K.V.: Real-Parameter Optimization with Differential Evolution. In: Proceedings of the 2005 Conference on Evolutionary Computation, Edinburgh, Scotland, September 2-5, 2005, pp. 888–895 (2005)
10. Sawai, H., Adachi, S.: Genetic Algorithm Inspired by Gene Duplication. In: Proceedings of the 1999 Congress on Evolutionary Computing, Washington DC, USA, July 6-9, 1999, pp. 480–487 (1999)
11. Serra, P., Stanton, A.F., Kais, S.: Method for global optimization. Physical Review, tome 55, 1162–1165 (1997)
12. Suganthan, P.N., et al.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, May 2005, AND KanGAL Report #2005005, IIT Kanpur, India (2005), http://www.dcs.ex.ac.uk/~dwcorne/cec2005/
13. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85, 317–325 (2003)
14. Van Den Bergh, F.: An Analysis of Particle Swarm Optimizers. Department of Computer Science, University of Pretoria, South Africa (2002)
15. Yasuda, K., Iwasaki, N.: Adaptive particle swarm optimization using velocity information of swarm. In: IEEE Conference on System, Man and Cybernetics, October 10-13, 2004, pp. 3475–3481. The Hague, Netherlands (2004)
16. Ye, X.F., Zhang, W.J., Yang, Z.L.: Adaptive Particle Swarm Optimization on Individual Level. In: International Conference on Signal Processing (ICSP), Beijing, China, August 26-30, 2002, pp. 1215–1218 (2002)

17. Yuan, B., Gallagher, M.: Experimental results for the special session on real-parameter optimization at CEC 2005, a simple continuous EDA. In: Proceedings of the 2005 Congress on Evolutionary Computation, Edinburgh, Scotland, September 2-5, 2005, pp. 1792–1799 (2005)
18. Zhang, W., Liu, Y., Clerc, M.: An adaptive PSO algorithm for real power optimization. In: APSCOM (Advances in Power System Control Operation and Management), S6: Application of Artificial Intelligence Technique (part I), Hong Kong, November 11-14, 2003, pp. 302–307 (2003)
19. http://clerc.maurice.free.fr/pso/
20. http://www.particleswarm.info

Evolution of Descent Directions

Alejandro Sierra Urrecho[1] and Iván Santibáñez Koref[2]

[1] Department of Computer Engineering, Escuela Politécnica Superior, Universidad Autónoma de Madrid, 28049 Madrid, Spain
Tel.: +34(91)497 2233; fax: +34 (91)4972235
alejandro.sierra@uam.es

[2] Bionics and Evolutionary Techniques Dept., Technische Universität Berlin, D-13355 Berlin, Germany
Tel.: +49(30)314 72663; fax: +49(30)314 72019
isk@bionik.tu-berlin.de

Summary. Estimation of distribution algorithms proceed by sampling new solutions from a probability distribution learnt in an evolutionary way. This involves keeping track of a population of candidate solutions and updating distribution parameters from the best of these candidates. We propose to substitute this population of solutions by one of descent directions. New solutions will no longer be sampled but interpolated along each direction in a deterministic way. Even when strong correlations between dimensions are present, sampling new directions from a product of independent one-dimensional Gaussian distributions is enough because covariances are captured by the directions. Despite its simplicity, our algorithm can address problems such as the rotated cigar function with state of the art performance and without any covariance calculation.

Keywords: descent directions, local search, evolution, estimation of distribution algorithm, continuous optimization.

1 Introduction

Efficient evolutionary optimization algorithms require the search distribution to stay within the evolution window [1]. Already early works in evolutionary computation realized the importance of the adaptation of the distribution from which new candidate solutions are obtained. The 1/5th success rule introduced by Rechenberg [2] has the aim to increase the progress rate of the optimization by adapting the variance (step size). When the success rate is too high, the variance increases. By contrast, when the success rate is below the optimal value, the variance decreases. This is an early example of the many adaptation strategies now available in the evolution strategy literature [3]. In genetic algorithms, one of the first attempts to skip the recombination operator is the population based incremental learning algorithm [4]. This algorithm samples new binary strings from probability distributions estimated for each bit out of the best individuals of the previous generation. This is one of the members of a family of algorithms known as estimation of distribution algorithms, in which

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 221–237, 2008.
springerlink.com

evolution is implemented through learning of and sampling from distribution probabilities [5]. In this paper we will be focusing on evolution strategies rather than genetic algorithms because we are interested in continuous minimization problems.

In evolution strategies, the estimation of distributions can be significantly simplified when the variables are independent. In this case, we are free to express the joint probability distribution as a product of one dimensional probability distributions. Gaussian distributions make things even easier, because all that is left to estimate is the mean and standard deviation of each dimension. This is precisely what the univariate marginal distribution algorithm (UMDA) [6] does for improving its Gaussian distribution. Means and deviations are updated out of the best solutions drawn from past Gaussian distributions.

Since this algorithm takes full advantage of the independence of the variables, it fails to work properly when correlations between them are present. When this is the case, drawing samples from one-dimensional Gaussian distributions falls short of providing an optimal way of sampling new points. A full fledged multivariate Gaussian distribution has to be learned in this case. As before, the most straightforward approach consists of estimating the covariance matrix out of the best representatives of the population. This is exactly the procedure carried out by the algorithm called EMNA [7].

Many other approaches are now available in the literature that exploit the information generated during the optimization process more effectively . For instance, the evolution strategy with cumulative step size adaptation (CSA-ES) [8] adapts the global step size by using the path traversed by the parent population over a number of generations. This algorithm has been extended with an adaptation of the covariance matrix. The new algorithm is called evolution strategy with covariance matrix adaptation (CMA-ES) [9]. This approach to adaptation is probably the most efficient for continuous minimization problems.

Another interesting approach is called IDEA [10] . This framework learns a graph or factorization of the joint probability distribution by optimizing a certain given metric. The graph construction is ended when adding new arcs does not lead to metric improvements. A related approach is called mixed Bayesian optimization algorithm (MBOA) [11]. This algorithm adapts a Bayesian network with decision trees for the estimation of the distribution probability of the selected parent population.

In this paper yet another possible approach is introduced. We call it evolution of descent directions (ED^2) . Instead of evolving a population of points in the domain of the goal function (solutions), we rather evolve a population of candidate descent directions along which we search for new points in the domain of the goal function. Each direction embodies a correlation between variables making use of only N parameters instead of the $N(N-1)/2$ necessary for the representation of a covariance matrix. Therefore, in this approach, the estimation of the covariance matrix is substituted by its evolution. One important question is how to assign fitness to each descent direction. Our proposal is to assess fitness by running a line minimization algorithm. In many cases it is enough to fit a

second degree polynomial to obtain the minimum on a line. This procedure has proved able to improve the population of directions and therefore the quality of the solutions. Broadly speaking, the algorithm works as follows:

1. An initial population of directions is generated randomly. The best point found so far is initialized at random. The following two points are iterated until a satisfactory best point is found.
2. Each direction is used to improve the best point by interpolation. Fitness is equal to the drop in function value.
3. New directions are sampled from a product of one-dimensional Gaussian distributions whose means and deviations are calculated out of the best current directions.

The organization of the paper is the following. Our algorithm is described in detail in section 2. Section 3 analyzes an illustrative experiment which highlights how our algorithm can solve correlated problems without the estimation of covariances. Section 4 reports on the experiments carried out with benchmark multivariate minimization problems. Although our algorithm requires in general more function evaluations than the best algorithm currently available, it is extremely efficient in terms of CPU time, due to its surprising simplicity and lack of second degree statistics. Our conclusions can be found in the last section of the paper.

2 The Algorithm: ED^2

Let us consider a multidimensional minimization problem, i.e., a real function $f : D \subset \mathbb{R}^N \to \mathbb{R}$ dependent on $\mathbf{x} \in \mathbf{D}$ whose minimum has to be found. Let us first describe the coding mechanism used by ED^2 so as to clarify the nature of the individuals evolved in our approach.

2.1 Coding Scheme

As mentioned above, our algorithm relies on a population of λ directions instead of points. More precisely, our individuals are straight lines in Euclidean space $\mathbb{R}^N$:

$$\mathbf{L}_i^g = \{\mathbf{x} + \beta \mathbf{w}_i^g \mid \beta \in \mathbb{R}\}, \tag{1}$$

where $\mathbf{w}_i^g$ is an N dimensional vector ($i = 1, \ldots, \lambda$), and $\mathbf{x}$ is the best N dimensional point found so far, i.e., the point with minimum function value found so far. Notice that this point has no subindex because it is shared by directions as explained next.

The basic idea behind our algorithm is to take advantage of efficient line minimization algorithms in order to improve the current best point. By means of one of these algorithms, each line $\mathbf{L}_i^g$ yields a new point $\mathbf{x} + \beta_i \mathbf{w}_i^{(g)}$ where

$$\beta_i = \min_{\beta} f(\mathbf{x} + \beta \mathbf{w}_i^{(g)}). \tag{2}$$

For quadratic functions, an interpolation procedure around the best point $\mathbf{x}$ is enough and most efficient. After successful interpolations, i.e., when $f(\mathbf{x} + \beta_i \mathbf{w}_i^{(g)}) < f(\mathbf{x})$, the best point is updated

$$\mathbf{x} := \mathbf{x} + \beta_i \mathbf{w}_i^{(g)}, \tag{3}$$

so that next lines can take advantage of the improvement. The fitness of each line is equal to the descent in the function value it gives rise to, i.e., each line is worth the function value before interpolation minus the value after interpolation. The best directions found in this way are used to update the one dimensional Normal distributions from which next directions will be sampled.

Let us summarize before closing this section. Our individuals are straight lines composed of directions and points. Only directions are sampled from Normal distributions, because points are found by interpolation on each line. The next subsection explains all this in detail.

2.2 The Algorithm

1. The size of the population (μ, λ) is chosen.
2. The initial one-dimensional Normal distributions are generated. Standard deviations are initialized at 1 (g is the generation number):

$$\sigma^{(g=0)} = (1, 1, \ldots, 1), \tag{4}$$

   and means are randomly chosen around zero:

$$\mathbf{m}^{(g=0)} = (m_1^{(0)}, m_2^{(0)}, \ldots, m_n^{(0)}) \sim \mathcal{N}(0, I). \tag{5}$$

3. The best point so far is initialized at random:

$$\mathbf{x} = (x_1, x_2, \ldots, x_n) \sim \mathcal{N}(0, I). \tag{6}$$

4. The following steps are repeated until an acceptable solution is achieved:
   a) λ new offspring directions are drawn from the current product of Gaussian distributions:

$$\mathbf{w}_i^{(g+1)} \sim \mathcal{N}(\mathbf{m}^{(g)}, \sigma^{(g)} \mathbf{1}^{\mathrm{T}}), \quad i = 1, \ldots, \lambda. \tag{7}$$

   b) The fitness value assigned to line $\mathbf{L}_i^{g+1}$ is the following difference in function value:

$$F(\mathbf{L}_i^{(g+1)}) = f(\mathbf{x}) - f(\mathbf{x} + \beta_i \mathbf{w}_i^{(g+1)}), \tag{8}$$

   between the best point so far $\mathbf{x}$ and the new one found by interpolation in its neighborhood (See section 2.3 for a detailed explanation.) Besides, the best point so far is updated after those interpolations which reduce function value (this is always the case for quadratic functions):

$$\mathbf{x} := \mathbf{x} + \beta_i \mathbf{w}_i^{(g+1)}. \tag{9}$$

c) The current one dimensional Normal distributions are updated from the μ best directions $\mathbf{w}_{i:\lambda}^{(g)}$ $(i = 1, \ldots, \mu)$ out of the λ we have just drawn from the previous distribution:

$$m_j^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} w_{i:\lambda,j}^{(g)}, \text{where } j = 1, \ldots, N \text{ and} \tag{10}$$

$$\sigma_j^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} (w_{i:\lambda,j}^{(g)} - m_j^{(g)})^2, \tag{11}$$

where $w_{i:\lambda,j}^{(g)}$ is component number j of direction $\mathbf{w}_{i:\lambda}^{(g)}$.

One of the most surprising features of ED^2 is its sampling mechanism. New directions are drawn from a product of independent one dimensional Normal distributions. The means and deviations of these distributions are all we have to estimate from the best candidate directions of each generation. Let us now explain in detail how fitness is assigned to each descent direction, a procedure which is at the heart of our algorithm.

2.3 Fitness Interpolation

At first, finding a means of assigning fitness to lines seems to be quite a complex task. However, it is not, because we only need to assess fitness locally, in the vicinity of each best point. This can be quickly done by means of a line minimization algorithm. Moreover, in the case of quadratic functions, a second degree polynomial interpolation is sufficient because the function is a parabola.

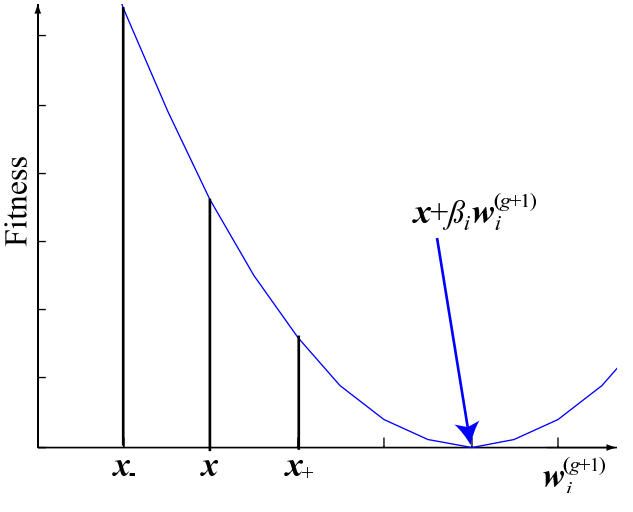


Fig. 1. Interpolation along direction $\mathbf{w}_i^{(g+1)}$

This procedure takes four function evaluations at the most. Figure 1 depicts the interpolation along direction $\mathbf{w}_i^{(g+1)}$ around the current best point $\mathbf{x}$, which leads in general to a better new point $\mathbf{x} + \beta_i w_i^{(g+1)}$. In fact, it takes just three function evaluations because the search lines pass through the best point found so far, whose fitness is already known.

More formally, what we have to find for each line $\mathbf{L}_i^{(g+1)}$ $(i = 1, \ldots, \lambda)$ is the coefficient β_i minimizing $f(\mathbf{x}+\beta\mathbf{w}_i^{(g+1)})$ in the vicinity of the best point $\mathbf{x}$ found so far:

$$\beta_i = \min_{\beta} f(\mathbf{x} + \beta\mathbf{w}_i^{(g+1)}). \tag{12}$$

The most reliable way of doing this is using a line minimization algorithm such as Brent algorithm [12], which is a clever combination of bracketing and interpolation. However, given that any line section of a quadratic function is quadratic itself, a simple second order polynomial interpolation method is sufficient and fastest. Only three function evaluations are needed in order to have a second order polynomial passing through three points $\mathbf{x}$, $\mathbf{x}_- = \mathbf{x} - \mathbf{w}_i^{(g+1)}$ and $\mathbf{x}_+ = \mathbf{x} + \mathbf{w}_i^{(g+1)}$. The coefficient of the minimum can be calculated by:

$$2\beta_i = -\frac{f(\mathbf{x}_-) - f(\mathbf{x}_+)}{2f(\mathbf{x}) - f(\mathbf{x}_-) - f(\mathbf{x}_+)} \tag{13}$$

Now, let us describe how to use the interpolated point $\mathbf{x} + \beta_i\mathbf{w}_i^{(g+1)}$ to assign fitness to line $\mathbf{L}_i^{(g+1)}$ $(i = 1, \ldots, \lambda)$. Instead of just using the value of the function value, we prefer to use the drop in function value along each line:

$$F(\mathbf{L}_i^{(g+1)}) = f(\mathbf{x}) - f(\mathbf{x} + \beta_i\mathbf{w}_i^{(g+1)}). \tag{14}$$

This is necessary for the following reason. After each successful interpolation, the best point is updated as the minimum of the parabola. The next line of the population interpolates around this new best point and, therefore, it automatically inherits this point's function value, no matter its quality as a descent direction. By using differences as we propose in equation 14, better fitness is assigned to those lines with more descent in function value, as expected.

2.4 Complexity of ED^2

One of the most attractive features of the evolution of directions is its simplicity. In contrast to many state of the art algorithms, the calculation here is dominated by function evaluations. In CMA-ES, the time necessary to compute the function value is small compared to the decomposition of the covariance matrix. Correspondingly, the learning sections of MBOA or IDEA also take most of the calculation time. In contrast, ED^2 is structurally similar to UMDA [6] after substituting points by lines. This means we are making use of the most economical self adapting mechanism available in the literature: drawing individuals from a product of independent one dimensional Gaussian distributions. Only means

and deviations need to be estimated, and no matrix inversion or covariance calculation is necessary, which burdens most other approaches. The only extra that has to be done here, in comparison to UMDA; is three function evaluations for each fitness assignment. In short, ED^2 is quite an efficient algorithm as compared with the state of the art estimation of distribution algorithms.

3 An Illustrative Experiment

In order to appreciate the advantages of our approach best, let us consider the cigar function in 10 dimensions:

$$f(\mathbf{x}) = x_1^2 + 10^4 \sum_{i=2}^{10} x_i^2, \tag{15}$$

together with a rotated version:

$$f_R(\mathbf{x}) = f(T\mathbf{x}), \tag{16}$$

where matrix T is a 20^o rotation involving coordinates x_1 and x_2. This second function poses serious trouble to evolutionary algorithms that sample from factorized probability distributions due to the correlation between coordinates. The point we wish to make clear in this section is that our algorithm, despite of sampling new descent directions from a product of one dimensional independent Gaussian distributions, it is still capable of spotting the right descent direction after solving the initial easiest portion of this minimization problem.

Figure 2 shows the best fitness so far versus the number of function evaluations for the rotated cigar function ($N = 10$) and three different optimization approaches: EMNA, UMDA and ED^2. This figure shows that sampling from one dimensional Normal distributions is sufficient when the population is integrated by descent directions (ED^2) instead of points (UMDA). The univariate marginal distribution algorithm gets trapped in a suboptimal solution because of sampling from independent Normal distributions. As expected, EMNA approaches the optimum in an almost linear way (in logarithmic scale) as a function of the number of evaluations. Our algorithm is the one which more rapidly solves the easiest part of this problem and then, after a plateau spent searching for the right descent direction, is capable of finally approaching the minimum along the rotated privileged direction.

Figure 3 shows the evolution in ED^2 for the rotated cigar function in 10 dimensions. Three quantities updated on a generational basis are shown: the norm of the mean of the μ best directions of each generation, the average norm of the μ best directions of each generation, and the norm of the variance estimated from the μ best directions. The proper direction is spotted after around 10000 evaluations in spite of the absence of covariance estimation. The two first approach each other when the population converges to one representative. This occurs after about 10^4 function evaluations, just when the variance begins to decrease. Although this number of evaluations may seem too high at first, we

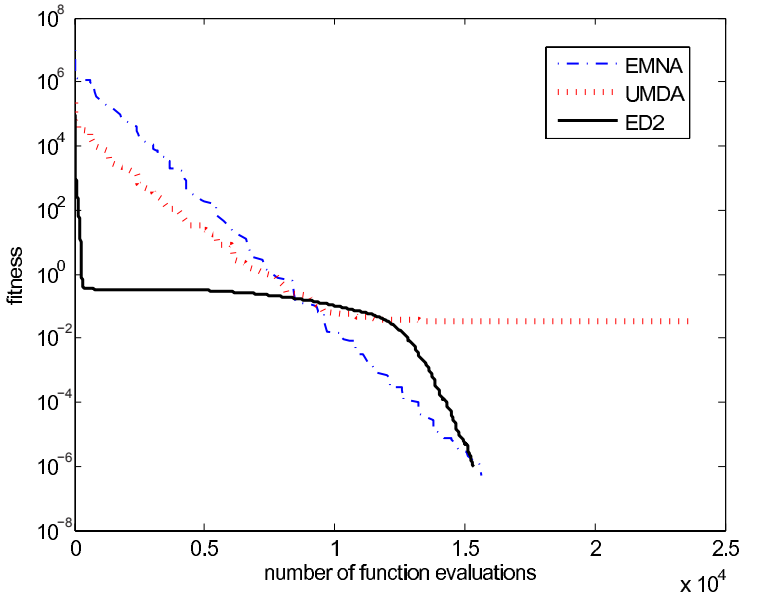


Fig. 2. Best fitness versus number of function evaluations for three different algorithms on the rotated cigar function (see text)

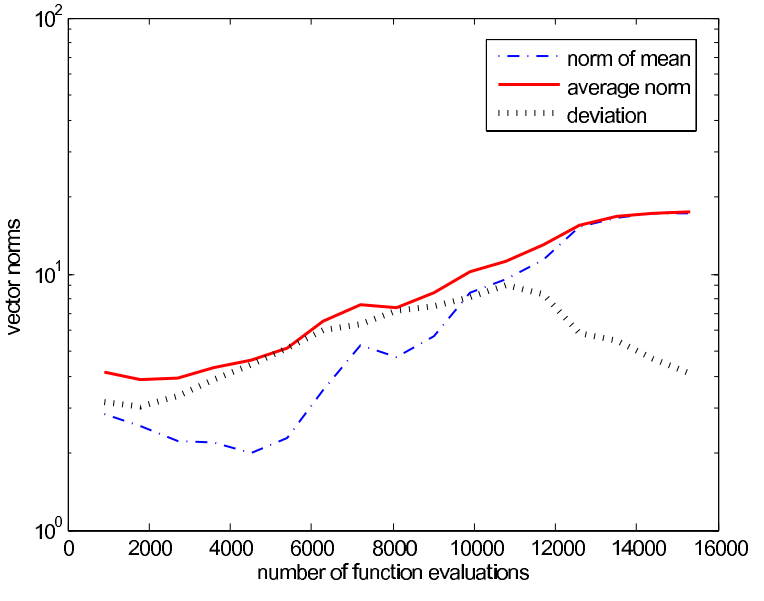


Fig. 3. Evolution in ED^2 for the rotated cigar function in 10 dimensions (see text)

remind the reader of the fact that in our approach no second order statistical estimation is carried out, thus making the algorithm quite efficient in CPU time.

4 Experimental Comparison

This section is devoted to the comparison of our algorithm with the state of the art estimation of distribution algorithms on a standard set of quadratic minimization problems (see Table 1). A lower condition value ($a = 100$) has been chosen to facilitate the comparison with a recent review [13]. All of our experiments are conducted for $N = 10$ and the results shown in this section correspond to the mean value of 20 executions for each function. Our initial population of directions is sampled from a product of Gaussian distributions centered around 0 and with standard deviations equal to 1. Remember that our individuals consist of lines (directions plus best point) instead of just points. The initial best point is set to the unit vector in all our experiments.

Table 1. Test functions and parameter values. The initial best point for all functions is the unit vector.

Name	Function	Parameters
Sphere	$f(\mathbf{x}) = \sum_{i=1}^{N} x_i^2$	
Cigar	$f(\mathbf{x}) = x_1^2 + a^2 \sum_{i=2}^{N} x_i^2$	$a = 10^2$
Ellipsoid	$f(\mathbf{x}) = \sum_{i=1}^{N} (a^{\frac{i-1}{N-1}} x_i)^2$	$a = 10^2$
Tablet	$f(\mathbf{x}) = (ax_1)^2 + \sum_{i=1}^{N} x_i^2$	$a = 10^2$

From our point of view, one of the nicest features of our algorithm is its simplicity: The only parameter we need to choose before running the algorithm is the size of the population. As expected, the behavior depends heavily on μ and λ. Our experiments show that μ can be chosen to be equal to the number of dimensions provided that selection pressure is increased significantly: ($\lambda = 10\mu$). Figure 4 makes this point clear. It shows the mean number of evaluations of the cigar function in 10 dimensions as a function of μ for three different values of λ. It is clear that we have to increase λ ($\lambda = 10\mu$) if we want $\mu = 10$ to give good results. This is quite a surprising result because it corresponds to only 20 generations, i.e., only 20 adaptations of the direction. Thus, in the following experiments a population ($\mu = 30, \lambda = 300$) has been used because its performance has less variance.

Table 2 compares the performance of our algorithm ($\mu = 30$, $\lambda = 300$) with other state of the art estimation of distribution algorithms. All of these algorithms have been described in the introduction ($(1 + 1) - ES$ is an evolution

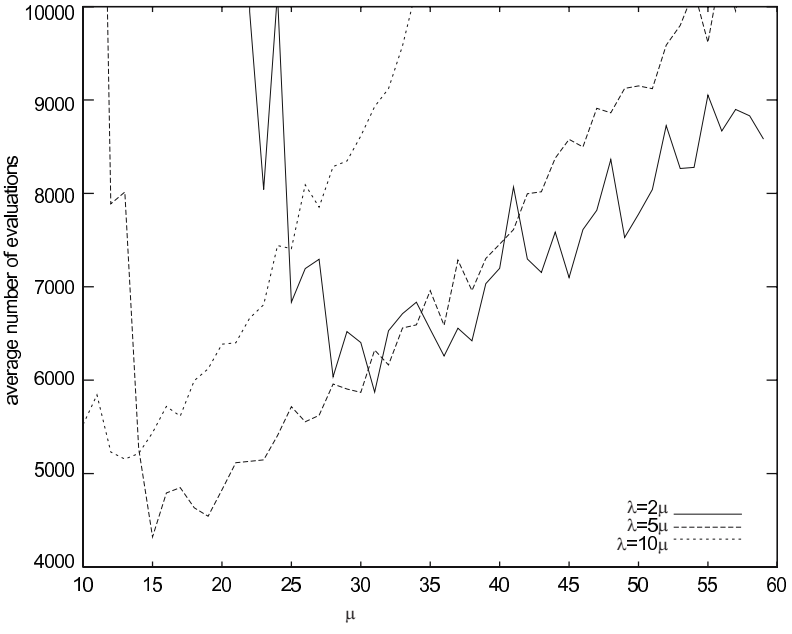


Fig. 4. Mean number of evaluations of the cigar function in $N = 10$ as a function of μ for three different values of λ

Table 2. Performance ratio comparison of some state of the art algorithms on a set of quadratic minimization problems. The number between parenthesis is the number of evaluations required by the fastest algorithm. The other numbers are ratios with respect to the figure in parenthesis.

Function	(1+1)-ES	CMA-ES	IDEA	MBOA	ED^2
Sphere	2	2.6	10	96.4	1 (682)
Ellipsoid	66	1 (4450)	1.6	14	2
Cigar	610	1 (3840)	4.6	12	2.2
Tablet	27	1 (4380)	1.7	14	1.2
Rot. ellipsoid	64	1 (4490)	13	1800	2.1
Rot. cigar	600	1 (3840)	38	2100	3.7
Rot. tablet	25	1 (4400)	6.8	910	2.7

strategy with the 1/5th rule). Surprisingly enough, ED^2 outperforms CMA-ES on the sphere function. MBOA's bad performance on this function has been attributed to its lack of overall variance estimation [13]. However, our algorithm performs extremely well on the sphere despite of its lack of overall variance estimation. The reason must lie in its sampling of directions instead of solution points.

Our algorithm is specially competitive for the non-rotated functions because our coordinate wise sampling mechanism is especially suited for non correlated

functions. Despite this simple mechanism, line minimization is capable of addressing all of the rotated functions without a severe increase in the number of function evaluations as can be checked in Table 2.

5 Scaling Properties of ED^2

One of the most attractive features of the evolution of directions is its simplicity. In fact, ED^2 is structurally similar to UMDA [6]: new directions are sampled from a product of one dimensional Gaussian distributions. Self adaptation amounts to recalculating means and standard deviations out of the best directions from previous generations. No covariance estimation procedure is necessary and the algorithm is extremely simple and fast. This simplicity contrasts with many state of the art EDAs which have very heavy learning algorithms designed to learn complex probability distribution functions. For instance, in CMA-ES, the time necessary to compute the function value is small compared with the decomposition of the covariance matrix. Correspondingly, the learning sections of MBOA or IDEA also take most of the calculation time.

The only extra work we have to perform in ED^2 as compared with UMDA, is running a line minimization algorithm. However, as we have discussed in Section 4, we use either parabolic interpolation, which only takes three function calls per

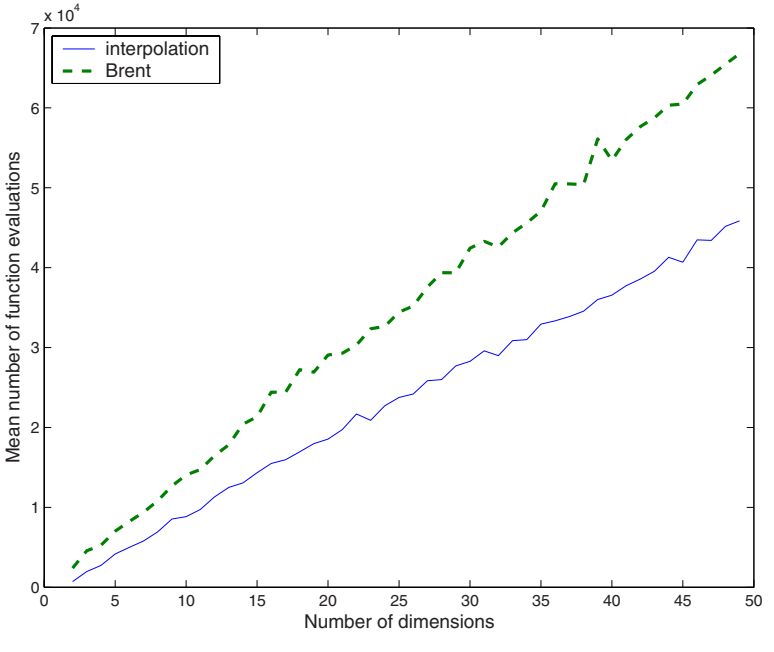


Fig. 5. Mean number of evaluations of the cigar function as a function of the number of dimensions. Two different line minimization algorithms are used: interpolation and Brent method.

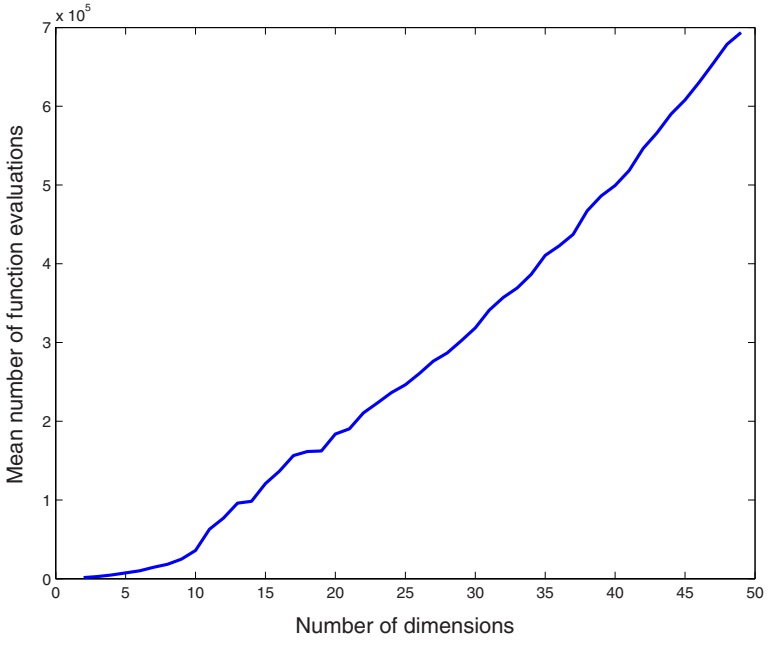


Fig. 6. Mean number of evaluations of the Rosenbrock function as a function of the number of dimensions

fitness evaluation, or Brent's method, with four function calls at most. In short, ED^2 is quite an efficient algorithm as compared with state of the art estimations of distribution algorithms.

Figure 5 shows how ED^2 performance scales with the number of dimensions N. Two curves are shown, one for each kind of line minimization algorithm: parabolic interpolation and Brent's method. Both curves represent the mean number of function evaluations as a function of the number of dimensions of the problem. The values shown are the averages of 20 executions of ED^2 with μ and λ set according to the optimal results for $N = 10$ obtained in Section 4. This population size depends on N, as expected, and also on the line minimization algorithm. More precisely, for $N = 10$, our experiments show that parabolic interpolation yields best results for $(\mu = 32, \lambda = 160)$, i.e., $(\mu = 3N, \lambda = 15N)$. Brent's line minimization algorithm is optimal for $(\mu = 51, \lambda = 306)$, i.e., $(\mu = 5N, \lambda = 30N)$.

Figure 5 clearly states that the mean number of function evaluations grows linearly with the number of dimensions as expected for quadratic functions. Brent's method is slightly slower than parabolic interpolation. This comes as no surprise, given that Brent's method together with the initial bracketing is more costly in terms of function evaluations than the elementary interpolation mechanism. Figure 6 represents the mean number of function evaluations of the Rosenbrock function as the number of dimensions grow. Here the behavior is quadratic, as corresponds to a non-quadratic function.

6 Conclusions

Many traditional minimization algorithms proceed by smart successive line minimizations [12]. More precisely, starting from a point $\mathbf{x}$ in N dimensions and a descent direction $\mathbf{w}$, a new point can be found by means of a one-dimensional minimization method along the line $\mathbf{x} + \beta\mathbf{w}$. Each method characterizes by the way in which new successive directions are chosen: the original axis, conjugate directions, etc. As expected, these choices are more efficient if the function's gradient is properly used. Our approach, which does not take advantage of gradient information, consists of estimating these directions by means of an evolutionary process. This involves keeping track of a population of lines, whose best representatives are used to sample new candidate search lines.

One of the most surprising features of our algorithm is that new descent directions are sampled from a product of independent one dimensional Gaussian distributions. This is arguably the most elementary adaptation mechanism available in the literature. In fact, it is known to fail when correlations between dimensions are present. However, when applied to the adaptation of lines, it gives rise to good results because correlated variables turn into independent ones when the proper direction is considered.

In order to judge which lines are best, a fitness evaluation process has to be designed. Our method is called fitness interpolation because for each direction $\mathbf{w}$ and current best point $\mathbf{x}$, β is found by having a second order polynomial pass through $f(\mathbf{x})$, $f(\mathbf{x} - \mathbf{w})$ and $f(\mathbf{x} + \mathbf{w})$. This procedure is an efficient one since it requires only three function evaluations. Given that any line section of a quadratic function is itself a quadratic function, second order interpolation is expected to perform as well but much faster than more clever line minimization algorithms. Although our experiments confirm this point, we are currently investigating other possibilities such as the popular Brent method [12], which is a rather smart combination of interpolation and bracketing. As expected, this alternative procedure outperforms simple interpolation for more involved functions such as the Rosenbrock function [13].

Apart from studying different line minimization methods, we are planning to generalize our algorithm in several ways. For instance, there is no need to restrict ourselves to linear descent directions. In example a neural network capable of learning non-linear search directions can be used, along which the algorithm interpolate new points. This is likely to outperform line minimization algorithms in the case of multimodal functions. In this paper we propose to set $\lambda = 10\mu$. The setting of λ, μ shall be examined. We expect to report soon on this ongoing project.

Acknowledgments

Alejandro Sierra would like to thank the members of the Bionik und Evolutionstechnik Laboratory of the Technical University of Berlin for their support during the months that it took to complete this work. This paper has been sponsored by a fellowship from the Spanish Ministry of Education and by the

Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TIN2004-07676-C02-02.

The work of I. Santibáñez has been supported by the project "Anwendung von Erkenntnissen der Bionik zur Funktions- und Strukturoptimierung von Schiffspropellern und anderen Gusskörpern" (01RW0307) granted by the Federal Ministry of Education and Research.

References

1. Rechenberg, I.: Evolutionsstrategie 94. Fromman-Holzboog Verlag, Stuttgart (1994)
2. Rechenberg: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
3. Beyer, H.G., Schwefel, H.P.: Evolution strategies: A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
4. Baluja, S., Caruana, R.: Removing the Genetics from the Standard Genetic Algorithm, Technical Report, Computer Science Department, Carnegie Mellon University, Pittsburg, PA, CMU-CS-95-141 (1995)
5. Mühlenbein, H., Paass, G.: From recombination of genes to the estimation of distributions I. binary parameters. Lecture Notes in Computer Science, vol. 1411, pp. 178–187 (1996)
6. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Optimization in continuous domains by learning and simulation of Gaussian networks. In: Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program, pp. 201–204 (2000)
7. Larrañaga, P., Lozano, J.A., Bengoetxea, E.: Estimation of distribution algorithms based on multivariate normal and Gaussian networks. Technical Report KZZA-IK-1-01, Department of Computer Science and Artificial Intelligence, University of the Basque Country (2001)
8. Ostermeier, A., Gawelczyk, A., Hansen, N.: Step-size adaptation based on non-local use of selection information. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) Parallel Problem Solving from Nature - PPSN IV, pp. 189–198. Springer, Berlin (1994)
9. 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)
10. Bossman, P.A.N., Thierens, D.: Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature - PPSN VI, pp. 767–776. Springer, Berlin (2000)
11. Ocenasek, J., Schwarz, J.: Estimation of distribution algorithm for mixed continuous-discrete optimizatino problems. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.-P. (eds.) Second Euro-International Symposium on Computational Intelligence, pp. 227–232. IOS Press, Slowakia (2002)
12. Fletcher, R.: Practical methods of optimization. John Wiley and Sons, New York (1987)
13. Kern, S., Müller, S.D., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms - a comparative review. Natural Computing 3, 77–112 (2004)

Source code of ED^2 in Matlab

In order to illustrate the ED^2-Algorithm, a sample code is presented in this section. The the comments refer to the equations in the above text.

```
function [feval,x,x_fit]=ed2(func_name)
%ED2 for unconstrained optimization
%Input:
% func_name - Name of goal function. Included functions in this file:
%               'kugel','cigar','tablette','ellipsoid','rosenbrock'
%Output:
% feval     - Number of goal function evaluations to meet stopping criteria
% x         - Best point found so far
% x_fit     - Goal function value at origin
% (c) 2006 A. Sierra and I. Santibaez

% Definition of constants and strategy variables
N = 10;                         % Dimension of problem
MU = 3*N;                       % Number of individuals to select and recombine
LAMBDA = 10*MU;                 % Number of offspring
MAX_FEVALS = 1e5;               % Maximal number of goal function evaluations
FITNESS_LIMIT = 1.0e-10;        % Stop limit for optimization

% Defintion of goal function and transformation
f.func = func_name;             % Name of fitness function
f.trans = false;                % Don't transformate input
f.T = [];                       % Transformation matrix
f.feval = 0;                    % Counter for number of fitness evaluations

% Initialization
z = zeros (LAMBDA, N);          % Perturbations
w = randn (LAMBDA, N);          % Array of offspring directions
w_fit = zeros(1,LAMBDA);        % Array of offspring fitness
w_point = zeros(1,N);           % Best point along offspring search direction
F = zeros(1,LAMBDA);            % Quality of each direction = drop in funct. value
sigma = ones (1, N);            % Array of std. dev. of selected directions Eq. 4
m = randn (1, N);               % Parent direction Eq. 5
x = initial(f.func,N);          % Origin of line search Eq. 6
[x_fit,f] = quali(f,x);         % Get fitness of starting point

% The ED2 algorithm
while (f.feval<MAX_FEVALS)
    z = randn(LAMBDA,N)*diag(sigma); % Generate perturbation
    w = m(ones(1,LAMBDA),:)+z;       % Generate new directions Eq. 7
    for l=1:LAMBDA                   % For each offspring
        [w_point,w_fit(l),f] = line_minimization(...   % Line minimization along
                                f,x,x_fit,w(l,:));  % offspring direction Eq. 12
        F(l) = x_fit-w_fit(l);       % Assign quality Eq. 8
        if F(l) > 0                  % If better
```

```
45              x_fit = w_fit(l);             % accept best point
46              x = w_point;                  % as new origin Eq. 9
47              if (x_fit < FITNESS_LIMIT)
48                  feval = f.feval;
49                  return
50              end
51          end
52      end
53      [F_s, idx] = sort (-F);          % Sort quality of offspring
54      m = mean (w (idx (1:MU), :));    % Recombine best MU offspring Eq. 10
55      sigma = std (z (idx (1:MU), :)); % Deviation of MU selcted
56                                       % perturbations Eq. 11
57  end
58  feval = f.feval;
59  return
60
61  function   [w_point,w_fit,fit] = line_minimization(fit,x,x_fit,w)
62  %Parabolic line search for one dimensional minimization
63  %Input:
64  % fit       - Structure containing goal function
65  % x         - Origin of line search
66  % x_fit     - Goal function value at origin
67  % w         - Direction describing the line to search
68  %Output:
69  % w_point   - Point found by line minimization
70  % w_fit     - Goal function value at w_point
71  % fit       - Structure containing goal function
72  left = -0.1; center = 0.0; right = 0.1;
73  [f_left,fit] = quali(fit,x+left*w);
74  f_center = x_fit;
75  [f_right,fit] = quali(fit,x+right*w);
76  denominator =  (center-left) * (f_center - f_right) ...
77                - (center - right) * (f_center - f_left);
78  numerator = (center-left)^2 * (f_center - f_right) ...
79            - (center-right)^2 * (f_center - f_left);
80  w_point = x + (center - 0.5 * numerator / denominator) * w;
81  [w_fit,fit] = quali(fit, w_point);
82  return
83
84
85  function [res,fit] = quali(fit,x)
86  %Wrapper for goal function
87  %Input:
88  % fit   - Structure containing goal function
89  % x     - Evaluation point
90  %Output:
91  % fit   - Structure containing goal function
92  % res   - Goal function at x
93      if fit.trans
```

```
94          x = x*fit.T';
95      end
96      fit.feval = fit.feval+1;
97      res = feval(fit.func,x);
98  return
99
100 function x0 = initial(func,N)
101 %Initial settings for each goal function
102 %Input:
103 % func - Name of goal function
104 % N    - Dimension
105 %Output:
106 % x0   - Start point for optimization
107 switch lower(func)
108     case {'kugel','ellipsoid','cigar','tablette'}
109         x0 = ones(1,N)/sqrt(N);
110     case 'rosenbrock'
111         x0 = zeros(1,N);
112     otherwise
113         disp('Unknown function');
114 end
115 return
116
117 % Goal functions
118 function res = kugel(x)
119 res = sum(x.^2);
120 return
121
122 function res = cigar(x)
123 a = 100;
124 res = x(1).^2+a.^2*sum(x(2:end).^2);
125 return
126
127 function res = tablette(x)
128 a = 100;
129 res = a.^2*x(1).^2+sum(x(2:end).^2);
130 return
131
132 function res = ellipsoid(x)
133 a = 100;
134 N = size(x,2);
135 fak = a.^( (1-N:0) ./(N-1));
136 res = sum((fak.*x).^2);
137 return
138
139 function res = rosenbrock(x)
140 res = sum(( 1 - x(1:end-1)).^2 + 100*(x(2:end) - x(1:end-1).^2).^2);
141 return
```

"Multiple Neighbourhood" Search in Commercial VRP Packages: Evolving Towards Self-Adaptive Methods

Kenneth Sörensen[1,2], Marc Sevaux[2], and Patrick Schittekat[3,4]

[1] Fellow of the Flemish Fund for Scientific Research, University of Leuven
Centre for Industrial Management, Celestijnenlaan 300a
3001 Leuven, Belgium
kenneth.sorensen@cib.kuleuven.be
[2] University of South Brittany
CNRS, FRE 2734, LESTER Centre de Recherche - BP 92116
F-56321 Lorient cedex, France
[3] University of Antwerp, Faculty of Applied Economics
Prinsstraat 13, 2000 Antwerp, Belgium
[4] ORTEC Belgium, 3150 Haacht, Belgium

Summary. All commercial packages for vehicle routing that the authors are aware of use a (meta)heuristic search procedure with several different neighbourhood structures. This paper attempts to answer the question why this is the case. As we will show, "multiple neighbourhood" search (MNS) is able to overcome the myopic behaviour of using only a single neigbourhood and is therefore more powerful. Also, MNS can be considered to be a very adaptable metaheuristic, which makes it especially suitable for the practical problems encountered in real life. We also point out that there is a need for the MNS applications used in commercial packages to evolve towards more self-adaptive systems.

Keywords: "Multiple neighbourhood" search, VRP, vehicle routing, commercial software.

1 Introduction

Vehicle routing — determining the order in which to visit a spatially distributed set of customers with a fleet of vehicles — is arguably one of the most useful and successful fields of operations research. As logistics and transportation generally constitute a sizeable proportion of the total cost of any company, it is not surprising that the efficient design of routes to visit customers for pickup or delivery can result in large savings. As a result, several companies have started to develop and market software packages for vehicle routing.

Although usually a closely guarded secret, we have through frequent informal contacts with several software developers gained some insight into the inner workings of the optimization engines of several commercial vehicle routing

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 239–253, 2008.
springerlink.com

packages.[1] From these contacts, we have learned several interesting facts. First, commercial software for vehicle routing is able to tackle problems that are an order of magnitude more complex than its academic counterpart. Second, all software packages for vehicle routing use heuristic search, based on (what would be called in academic circles) a systematic changing of neighbourhood structures during the search. In this paper, we will refer to such heuristic search as "multiple neighbourhood" search or MNS. MNS is not a new metaheuristic (hence the quotes, which we will drop from now on), but rather a concept underlying several optimization approaches.

In this paper, we look into some of the differences between vehicle routing in real life and vehicle routing in the academic world (section 2) and discuss the complexity of real-life vehicle routing problems (section 3). We look into the use of the concept of multiple neighbourhood search in the academic world (section 4) and in industry (section 5) and attempt to find out the reasons for using MNS in commercial vehicle routing software (section 6). We also argue that there is a strong need for commercial vehicle routing software to become more self-adaptive (section 7). In section 8, we draw some conclusions.

2 Academic Versus Real-Life Vehicle Routing

The importance of vehicle routing, as well as its complexity, has forced many companies to use software packages to manage these problems. In the past, software for vehicle routing was usually tailor-made for a specific company, but this has changed and in recent years several companies have started to develop and market commercial software packages for vehicle routing. These are either sold as stand-alone packages or integrated into large ERP (Enterprise Resources Planning) packages such as SAP. Typically, this software is put to work in highly complex and dynamic environments. Moreover, real-life vehicle routing problems are generally large (often several thousands of customers) and should be solved quickly.

Academic research on vehicle routing on the other hand has focused mainly on problems that greatly simplify the complex reality of real-life vehicle routing. Research on routing problems builds on research into a number of very basic problems (most notably the simple capacitated vehicle routing problem, the mathematical formulation of which can be found in Figure 1) and adds complexity to these simple models. In research on more complex problems, generally only a very small number of additional features is added. Typical problems that are investigated can be found for example in the categories of problems with *time windows*, *pick-up and delivery* problems or *heterogeneous fleet* problems. Often,

[1] Given the strategic importance, most companies are understandably reluctant to share the exact functioning of the optimization engines used in their software. Several of them indicated that they would be unwilling to supply this information if their name would end up in the open literature. Therefore, we have opted not to mention the names of the companies or their products.

$$\min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{v=1}^{V} c_{ij} x_{ijv} \tag{1}$$

$$\text{s.t.} \sum_{v=1}^{V} z_{iv} = 1 \qquad \forall i, i \neq 0 \tag{2}$$

$$\sum_{v=1}^{V} z_{0v} = V \tag{3}$$

$$\sum_{i=1}^{N} a_i z_{iv} \leq b_v \qquad \forall v \tag{4}$$

$$\sum_{j=0}^{N} x_{ijv} = \sum_{j=0}^{N} x_{jiv} = z_{iv} \qquad \forall v, i \tag{5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijv} \leq |S| - 1 \qquad \forall v, S \subset N \tag{6}$$

$$z_{iv} \in \{0, 1\} \qquad \forall i, v \tag{7}$$

$$x_{ijv} \in \{0, 1\} \qquad \forall i, j, v \tag{8}$$

Symbols: c_{ij}: distance between customers i and j; V: number of vehicles; b_v: capacity of vehicle v; a_j: demand of customer j; x_{ijv}: binary decision variable, 1 if vehicle v travels from customer i to customer j and 0 otherwise; z_{iv}: binary decision variable, 1 if customer i is visited by vehicle v

Fig. 1. Capacitated vehicle routing formulation [8]

problems are relatively small, generally not exceeding a few hundred customers and large computing times are allowed.

The difference between academic and real-life vehicle routing is illustrated in Figures 2 and 3, that show a graphical representation of the solution to an academic vehicle routing problem (the capacitated vehicle routing problem) and a screen shot of a commercial software package displaying the solution to a real-life routing problem. It is clear that real-life routing problems are several orders of magnitude more complex than their academic counterparts. Although there is an increasing academic focus on so-called "rich" vehicle routing problems (that incorporate more complex constraints and objectives), they have not in any way caught up with the whole complexity of real-life routing problems.

Another interesting difference between academic and real-life vehicle routing is that academic vehicle routing models are usually relatively easily expressed as a mathematical programming problem. In most cases, a relatively concise mixed-integer linear programming formulation can be given, similar to the one shown in Figure 1. This is not the case for real-life vehicle routing problems: mathematical programming formulations for such problems are generally not explicitly developed as the cost of doing would not outweigh the benefits. This

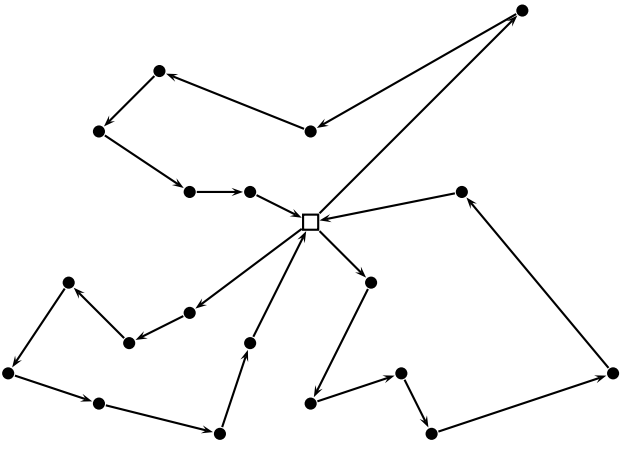


Fig. 2. A typical "academic" VRP solution

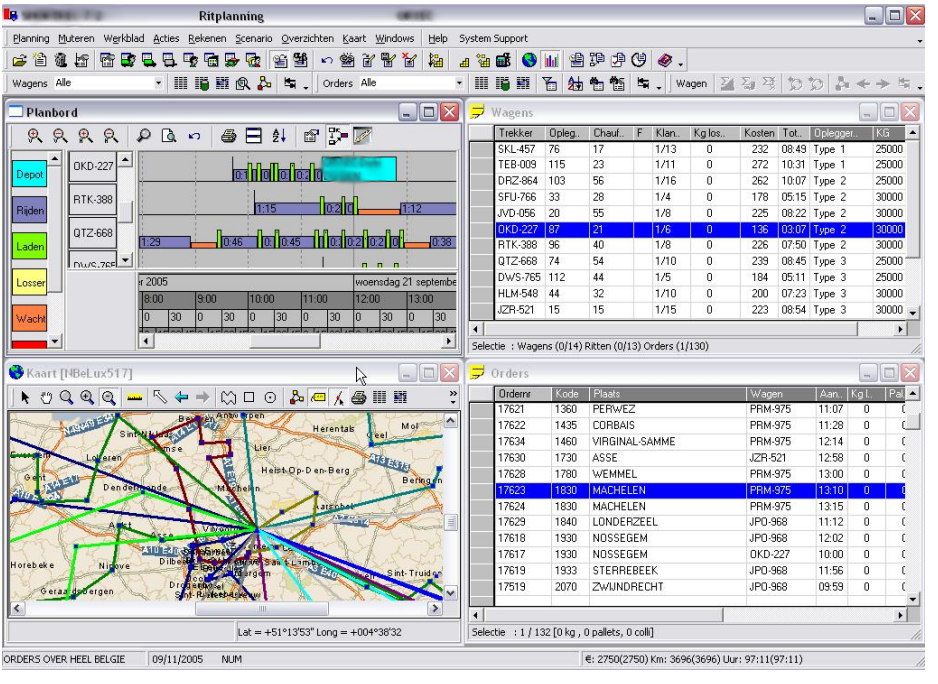


Fig. 3. A real-life VRP solution

limits the usefulness of black-box integer programming solvers, such as CPLEX [6] in real-life vehicle routing (although such libraries may of course be used as part of a more complex solution process).

3 The Complexity of Real-Life Vehicle Routing

One of the most important characteristics of commercial vehicle routing software is that it needs to be able to model virtually all types of difficult characteristics that any problem might have. Otherwise, the software runs the risk of using much of its usefulness. For example: if some of the orders have to be picked up at one location and delivered at another, then the software needs to be able to model this. If customers require service to take place within certain time limits, the software needs to cater for time windows. In Figure 4, a non-exhaustive list is shown of different characteristics that a real-life vehicle routing problem may exhibit. We have divided them into five categories: customer characteristic, driver characteristics, vehicle characteristics, route characteristics, and other/general characteristics.

Real-life vehicle routing problems are not stand-alone problems, but have an impact on other decisions taken in the company's supply chain. Vehicle routing packages should therefore seamlessly integrate with the company's ERP system. Ideally, it should be able to assist in taking such decisions as determination of drop size in a vendor-managed inventory (VMI) environment, trailer drop-off location, driver assignment, etc. Currently, commercial VRP packages rarely provide the necessary tools for making these decisions.

Another difference between commercial and academic vehicle routing software is the need for a functional graphical user interface, allowing the dispatcher to easily interact with the system. This includes making changes to the final solution while leaving some parts untouched, adding/deleting orders at the last moment, changing drivers from one vehicle to another, etc. Commercial vehicle routing software also needs to be able to get its distance and time estimates from a reliable source, e.g. a map provided by a digital cartographer such as Navteq or Teleatlas. Preferably, the software should provide interfaces to several of such data sources, as the company might already own one of them and may be unwilling to change. Also, interfacing to in-vehicle communication systems is a much desired feature.

In June 2006, ORMS Today ran a survey of 20 packages for vehicle routing from 17 companies [9]. It should be mentioned that this survey presents information provided by the companies themselves and should therefore be interpreted with some caution. The survey underlines the fact that commercial vehicle routing software is used in a large variety of environments and traditionally fulfils a number of different functions: sequencing stops, scheduling stops and assigning stops to drivers. It also discusses several recent developments in commercial VRP software, such as the integration with the company's ERP package and the ability to communicate with the drivers in order to be able to perform last-minute route changes. Unsurprisingly, a conclusion of the survey is that specialization, i.e. gathering experience in a certain sector, may give the software a competitive advantage. Another conclusion of the ORMS Today survey is that the size and complexity of real-life routing problems have rendered them intractable to solve using exact methods. Designers of commercial routing packages have therefore resorted to using heuristics.

- *Customer characteristics*
  - time windows (soft/hard)
  - pick-up/delivery/both
  - special requirements with respect to driver or vehicle visiting
  - product type(s) and quantity demanded
  - ...
- *Driver characteristics*
  - hours available
  - required resting times
  - ability to drive some vehicles and others not
  - legal regulations
  - ...
- *Vehicle characteristics*
  - heterogeneous fleet (different types/sizes of vehicles)
  - some vehicles may have multiple compartments for different products
  - special equipment (some vehicles may have cranes or loading equipment, others not, ...)
  - not all vehicles start and/or end at the depot
  - special licence required to drive a certain type of vehicle
  - different cost structures
  - ...
- *Route characteristics*
  - travel time between two points may change over time (e.g. longer during rush hours)
  - some vehicles may not be able to traverse certain routes or make certain turns, as a consequence sometimes extra decisions have to be made, for example where to drop off a trailer in order to be able to visit less accessible clients
  - ...
- *Other/general characteristics*
  - completely different routing problems (buses, taxis, garbage collector cars, transport of disabled people, ...)
  - multiple heterogeneous depots (e.g. carrying different products and having different stock levels for each product)
  - stochasticity
  - dynamic information
  - different/multiple objectives (cost, balance between route lengths)
  - ...

Fig. 4. Real-life vehicle routing problems possibly exhibit many different characteristics

A remarkable result of our contacts with developers of vehicle routing software is that a large number of commercial VRP packages (all packages that the authors are aware of), attempt to improve upon solutions (obtained by initial heuristics) by using a relatively large arsenal of local search improvement heuristics, based

around different neighbourhoods. As mentioned, we will use the term multiple neighbourhood search (MNS) to describe these types of metaheuristics.

4 MNS in Academic Vehicle Routing

All metaheuristics use mechanisms for intensification (focussing the search on a specific region of the search space) and diversification (directing the search towards previously unexplored regions of the search space). As mentioned, multiple neighbourhood search is not a metaheuristic in itself, but rather a property of metaheuristics that are based on the principle of systematically exploring more than one type of neighbourhood. MNS is a concept that is prevalent in a number of metaheuristics, most notably variable neighbourhood search (VNS).

VNS, a metaheuristic technique proposed some years ago by [14,10,11], has quickly gained a widespread success and a large number of successful applications have been reported [12,13]. Several variants of VNS have been described, such as variable neighbourhood descent, reduced variable neighbourhood search, etc., all of which have a slightly different way of performing the search. Most implementations of VNS use a sequence of nested neighbourhoods, $\mathcal{N}_1$ to $\mathcal{N}_{k_{\max}}$, in which each neighbourhood in the sequence is "larger" than its predecessor, i.e. $\mathcal{N}_k \subset \mathcal{N}_{k+1}$.

Pseudo-code for a basic version of VNS is given in algorithm 1.

Algorithm 1. Basic variable neighbourhood search

```
initialise: find an initial solution x, k ← 1;
repeat
    shake: generate a random solution x' ∈ N_k(x);
    local search: x' → x'';
    if x'' is better than x then
        x ← x'' and k ← 1 (centre the search around x'' and search again with a
        small neighbourhood);
    else
        k ← k + 1 (enlarge the neighbourhood);
    endif
until k = k_max ;
```

Several researchers have applied some form of variable neighbourhood search to (more or less academic) vehicle routing problems. Two examples are [3], in which a variable neighbourhood search algorithm for the vehicle routing problem with time windows is developed, and [7], who present a VNS approach to the vehicle routing problem with backhauls.

It is important however to note that the concept of *multiple* neighbourhood search is broader than that of *variable* neighbourhood search and that a large number of non-VNS approaches use different neighbourhood types. On the contrary, multiple neighbourhood search is overwhelmingly present in the list of

best-performing approaches for any vehicle routing problem. In their survey article on metaheuristics for the vehicle routing with time windows, [1] list the neighbourhoods used by different methods. Of the 14 tabu search methods listed, 10 use more than one neighbourhood type. More surprisingly, out of 17 genetic algorithms — a method not traditionally associated with the use of neighbourhoods — a majority (11) use more than one neighbourhood type to improve upon the solutions generated by crossover. Some approaches even use a very large number of neighbourhoods, such as the evolutionary algorithm by [15] for the capacitated VRP, that uses nine different neighbourhood types.

5 MNS in Commercial VRP Software

Although most academic software for vehicle routing uses more than one neighbourhood type, the use of several neighbourhood types is usually not considered to be the main mechanism that ensures intensification and diversification. A large majority of the academic software for vehicle routing, uses other mechanisms to achieve this, such as memory structures (tabu search algorithm) or evolutionary operators such as crossover and random mutation (genetic algorithms).

A surprising fact therefore, when viewed from a scientific viewpoint, is that none of the programs in our survey used any of the more complicated techniques that are commonplace in present-day metaheuristics. These techniques, such as memory structures and random perturbations, are hardly ever used, let alone complicated operators like crossover or mutation. In other words, the use of multiple neighbourhood search is not only prevalent in commercial vehicle routing software, it constitutes the main search mechanism to supply intensification and diversification. In the next sections, we attempt to explain the reasons for the popularity of the multiple neighbourhood concept and the lack of advanced metaheuristics techniques in commercial software packages.

Several types of heuristic search strategies can be found in commercial software packages:

- Construction heuristics (savings, clustering,...). The savings-based construction heuristic by [2] still seems to be very popular, probably due to the fact that it is easy to understand and implement and can be adapted to take many different constraints into account.
- Manipulate one or more stops within one route
- Manipulate one or more stops between routes.
- Replace/swap one route or more between vehicles.
- Equalize route lengths.
- ...

Most of the optimization work is done in so-called local improvement heuristics, that operate by manipulating stops within or between routes. Basically, such heuristics attempt to improve a solution by making one atomic change (called a *move*) at a time. In Figure 5, we show several types of such moves. We should be careful to distinguish between a *move type* and a specific *move* applied to specific location in a specific solution. Similarly, we distinguish between

a *neighbourhood*, i.e. the set of all solutions that can be reached in a single move starting from a given solution and a *neighbourhood structure*, which is essentially the same as a move type.

The *remove–insert* move type removes one stop from a route and inserts it at another location in the same route or in a different route. The *2-opt* move type

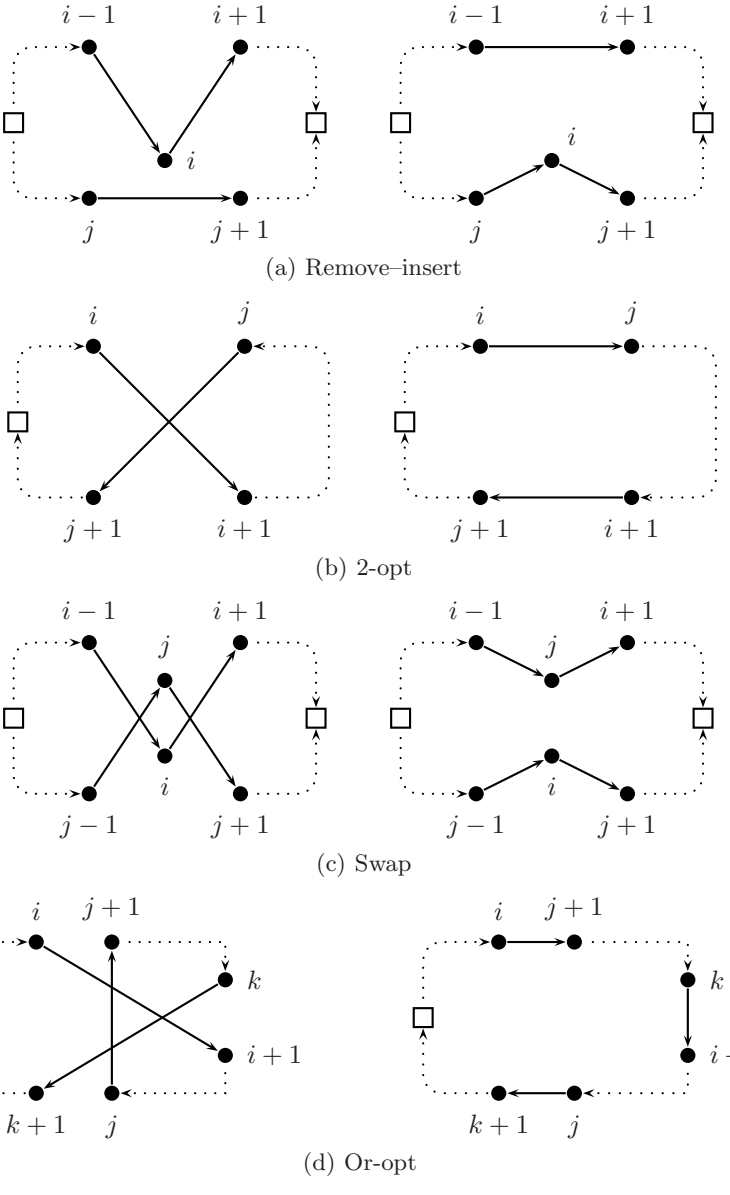


Fig. 5. Different move types

removes two edges from a solution and reconnects the solution in a different way. This move type is the simplest of a family of move types called k-opt, where k is the number of edges to remove. As k increases, so does the number of possible ways to reconnect and hence the complexity of this move type. A *swap* exchanges the location of two customers in a route. *Or-opt* is a specific case of 3-opt, in which, three edges are removed and reconnected in such a way that the orientation of the middle part of the route remains the same. It can be easily seen that this is not the case in 2-opt, where the orientation of a part of the route changes.

6 Reasons for Using MNS

In this section, we investigate why MNS is so popular in commercial VRP packages, as opposed to more complicated metaheuristic strategies. We see two main reasons for this. Firstly, due to its large arsenal of neighbourhoods, MNS is able to easily overcome the myopic behaviour of an approach that uses a single neighbourhood. Secondly, the neighbourhood operators used in a MNS approach can be easily rearranged as the problem size and complexity requires.

The use of MNS in commercial routing software has generally not been a deliberate decision. Rather, most of the software packages that we have encountered seem to have organically developed into the powerful tools that they are at this moment, adding neighbourhood structures and operators as this was required by the different problems that needed to be solved.

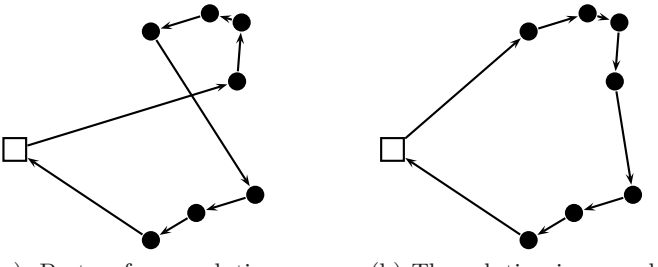
6.1 Overcoming the Myopic Behaviour of a Single Neighbourhood

When watching a local search approach improve upon a VRP solution, it is often remarkable to see how an improving move that is obvious to a human observer, is not discovered by the metaheuristic or only after several (often random) perturbation moves. Such behaviour is clearly unacceptable for expensive commercial VRP packages: if a dispatcher notices upon first inspection of the solution that the software has missed several important opportunities for improving the solution, confidence in the software will drop considerably. The reason for this myopic behaviour is quite often the fact that the metaheuristic is stuck in a so-called *local optimum*: the move required to improve the solution cannot be performed, because it does not belong to the move type that is used, and each of the moves in the neighbourhood would lead to a deterioration of the quality. Metaheuristic optimization strategies can be seen as essentially strategies to allow the metaheuristic to escape from such local optima. Instead of relying on advanced metaheuristic mechanisms such as random perturbations (iterated local search) or memory structures (tabu search), or crossover and mutation (evolutionary algorithms) however, multiple neighbourhood search proceeds in this case by using a different type of neighbourhood, which might contain the required improving move.

One of the basic principles behind the multiple (and variable) neighbourhood search technique is exactly the fact that a local optimum in one neighbourhood

is not necessarily a local optimum in another. In other words, when the search is stuck in a local optimum, it may suffice to switch to another neighbourhood type to be able to escape from it. Besides the advantage of finding a better solution, this strategy has the additional advantage of not (or with a much smaller probability) leaving obvious chances for improvement.

This is illustrated in Figure 6. Suppose we are optimizing this simple VRP tour using a local search algorithm that uses a remove–insert move (i.e. move a customer to another location in the solution). Some investigation shows that this solution cannot be improved by the remove–insert move type. Any solution in the neighbourhood of this one is worse, and therefore the search is stuck in a local optimum. A 2-opt move however (remove two edges and reconnect the solution) has no problems with this solution and will find the much better solution depicted in Figure 6(b) in one move. Although the possibility for improvement of the solution in Figure 6(a) is immediately obvious, it is far from certain that metaheuristic techniques such as memory structures or random perturbations will help to find this solution. Only the presence of the "right move" is certain to help.



(a) Part of a solution unimprovable by a remove-insert move type

(b) The solution improved with a 2-opt move

Fig. 6. Myopic behaviour of the move neighbourhood that the 2-opt neighbourhood is able to overcome

6.2 Flexibility and Adaptability to Different Problems

Vendors of commercial VRP packages face a number of challenges that are different from the ones faced by academic researchers. One of these is the inability to develop completely new methods each time a new problem is encountered. This would generally require rewriting large portions of the code base of the algorithms used in the software packages for every single customer and would quickly render the operations far less profitable. On the other hand, a black box optimizer that does not take the problem structure into account at all, does not provide the solution quality required.

We believe that it is for these reasons that designers of commercial routing packages have opted for an approach that supplies a (relatively large) set of components. These components can generally be divided into two categories:

constructive heuristics and improvement heuristics. Whereas the former construct a good initial solution using a heuristic construction rule, the latter use local search to improve upon a solution.

Consultants of the software vendor can then use these components to quickly and often effectively create a solution method partially tailored to the specific needs of the company using the software. The way in which these components are combined and exactly how they work is of course specific to the software package, but it can be stated that they all use different neighbourhood structures to search for better solutions and can hence be considered to be multiple neighbourhood search.

Having a large library of search modules and being able to combine them to suit the needs of the specific client, allows the routing software vendor to adapt to the different environments in which the software may be installed. This includes adapting it to specific constraints or objectives (e.g. some heuristics may perform well if the problem has time windows and otherwise not), but also adapting it to completely different problems, such as school bus routing problems or dynamic routing of ambulances. It is through this process that the program is adapted to the computational resources and required solution times imposed by the customers. Some companies may require their solutions after a few seconds, whereas others require them only after one night of calculation time.

The drawback of this approach, however, is that it requires a lot of manual intervention from the part of the software vendor. Adapting the software to the specific requirements of a customer can only be done by skilled consultants, that know both the software and the clients' environment very well.

7 Towards Adaptive MNS for Real-Life Vehicle Routing

One of the main problems with the approach described above is that the implementation of a commercial routing package typically requires quite a lot of manual work to be done. In some areas, this situation has dramatically improved over the last few years. Data import and export, for example, are typical areas in which customized modules would be written in order for the routing package to be able to communicate with the clients' data warehouse. Recently, however, through increased standardization and the use of XML technologies, data integration has become less of a problem and requires less and less manual intervention.

An area where there is far less progress is exactly in the mentioned manual tuning of the optimization approach. To date, this step in the roll-out of the software package still needs to be done by expensive consultants. There is a strong need for far more automation in this field, which naturally would require the algorithms to be self-adaptive. It can be envisaged that future implementations of commercial routing packages would include some kind of "hyper"-algorithm that would tune the configuration of the software before or during the actual optimization. This can be done off-line (using e.g. a set of test data) or on-line (while optimization is going on, using the actual data that is being processed).

Ideally, an off-line algorithm that determines the configuration should be able to do this based only on a number of historical data sets, perhaps updated with a prediction of future changes to the data (e.g. expected growth of problem size), and some maximum solution time. Based on this information, the configuration module should be able to determine the ideal configuration to produce a good solution to a problem similar to those in the test set, in a computation time that is within the bounds set by the decision maker. An on-line algorithm can do the same, but should also be capable of updating the configuration while the optimization is running. One can expect on-line algorithms to be useful for settings in which the allowed computation time is rather large (e.g. 12 hours) and off-line algorithms when the allowed computation time is rather small (e.g. a few minutes).

We expect that an adaptive hyper-algorithm will use the lower-level heuristics based on the different neighbourhoods as "building blocks". For each specific problem it will construct an appropriate solution method by changing:

- the order in which the heuristics are used;
- the frequency with which the heuristics are used;
- the amount of time each heuristic is allotted.

Ideally, the hyper-algorithm should be able to automatically tune the software package without any user intervention. This has the advantage of making installation considerably faster, and moreover, the VRP package will be able to adapt itself to changing requirements (e.g. larger data sets as a result of company expansion) without requiring manual intervention.

Research into such hyper-algorithms is rather scarce at the moment. A promising trend is a new type of heuristic coined *hyperheuristic*, that has been proposed [4,5] recently. A hyperheuristic can be informally defined as *a heuristic that selects heuristics*. Contrary to an ordinary or a metaheuristic, a hyperheuristic does not search in a space of *solutions*, but rather in a space of *heuristics*. Ideally, a metaheuristic controls a set of (more or less problem-specific) low-level heuristics, without using any problem-specific information. The hyperheuristic, in other words, is completely unaware of the problem that is being solved under its supervision and only uses information reported to it by the underlying heuristics such as the CPU time required, and the objective function value improvement obtained by a certain heuristic.

It is claimed that a hyperheuristic may approach the speed and solution quality of an approach that uses problem-specific information while only using cheap and easy-to-implement low-level heuristics. The full potential of hyperheuristics has yet to be established, but they certainly provide a valuable direction for future research in this area. However, hyperheuristics are at this moment not without their drawbacks. For many hyperheuristics, a significant amount of parameter tuning is required in order for them to perform adequately. It might seem tempting to regard this as a motivation for the development of procedures to automatically determine the parameters of the hyperheuristics. However, such hyper-hyperheuristics might be susceptible to the same problem, i.e. they might themselves require parameter tuning. It is clear that an infinite loop of ever

higher-lever heuristics tuning the parameters of a lower-level heuristic is not a desirable thing. Instead, the quest for good hyperheuristics should probably be aimed at developing methods that have little or no parameters and hence require no tuning.

The current state of the art in hyperheuristics is such that a large number of approaches exist, using different methods to select lower level heuristics. Many of these approaches include a large amount of randomness, but intelligent strategies are common too. A disadvantage of most approaches is that they have not been tested in real-life situations, and that the performance of these approaches under the full complexity of reality still remains a question.

It is the authors' belief that a high-level approach to determine the configuration of the optimization algorithm must make use of the relationship between the problem structure and the quality of a heuristics optimization strategy. In other words, a hyper-approach can only work well if there is an understanding of why a certain heuristic works better on a certain problem type than on another one. Armed with this knowledge, a hyperheuristic can then efficiently and effectively determine the optimal configuration of the underlying search heuristics. It can be argued that research into heuristics has always been a rather empirical domain and that the quality of a metaheuristic approach has only been judged based on its actual performance on a set of test problems. Recently, some research into the mechanisms that determine the effectiveness of a (meta)heuristic optimization approach, have been undertaken (see e.g. [16]), but a lot more research is needed in this domain, especially with respect to more complex problems.

8 Conclusions

In this paper, we have argued that the fact that all commercial packages for vehicle routing use some form of multiple neighbourhood search is due to two factors, both related to the complexity of real-life problems. On the one hand, an approach that uses many neighbourhoods simultaneously may overcome the myopic behaviour of one that uses only a single neighbourhood. Secondly, supplying a large arsenal of local search strategies based on different neighbourhoods allows the consultants of the routing software vendor to flexibly adapt the software to the specific requirements of each client. We have further argued that there is a strong trend towards more self-adaptive approaches, that overcome the need for manual parametrization of the software package. A promising research domain is that of *hyperheuristics*, but a lot more research is needed in this area.

References

1. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part II: Metaheuristics. Transportation Science 39, 119–139 (2005)
2. Clark, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581 (1964)
3. Cordone, R., Wolfer-Calvo, R.: A heuristic for the vehicle routing problem with time windows. Journal of Heuristics 7, 107–129 (2001)

4. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling PATAT 2000. LNCS, pp. 176–190. Springer, Heidelberg (2001)
5. Cowling, P., Kendall, G., Soubeiga, E.: A parameter-free hyperheuristic for scheduling a sales summit. In: MIC 2001 – Proceedings of the Metaheuristics International Conference, Porto, pp. 127–131 (2001)
6. CPLEX Optimization, Inc., Suite 279, 930 Tahoe Blvd., Bldg, 802, Incline Village, NV 89451-9436. Using the CPLEX Callable Library (1995)
7. Crispim, J., Brandao, J.: Reactive tabu search and variable neighborhood descent applied to the vehicle routing problem with backhauls. In: MIC 2001 – Proceedings of the Metaheuristics International Conference, Porto, pp. 631–636 (2001)
8. Fisher, M.L., Jaikumar, R.: A generalized assignment heuristic for solving the vrp. Networks 11, 109–124 (1981)
9. Hall, R.: The 2006 vehicle routing survey. ORMS Today 33(3) (June 2006)
10. Hansen, P., Mladenović, N.: Variable neighborhood search for the p-median. Location Science 5, 207–226 (1997)
11. Hansen, P., Mladenović, N.: An introduction to variable neighborhood search. In: Voss, S., Martello, S., Osman, I., Roucairol, C. (eds.) Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 433–458. Kluwer, Boston (1999)
12. Hansen, P., Mladenović, N.: Industrial applications of the variable neighbourhood search metaheuristic. In: Decisions and Control in Management Science, pp. 261–274. Kluwer, Boston (2001)
13. Hansen, P., Mladenović, N.: Variable neighbourhood search: Principles and applications. European Journal of Operational Research 130, 449–467 (2001)
14. Mladenović, N.: A variable neighborhood algorithm – a new metaheuristic for combinatorial optimization. In: Optimization Days, p. 112 (1995)
15. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 31, 1985–2002 (2004)
16. Watson, J.P., Howe, A.E., Whitley, L.D.: Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. Computers and Operations Research 33, 2623–2644 (2006)

Automated Parameterisation of a Metaheuristic for the Orienteering Problem

Wouter Souffriau[1,2], Pieter Vansteenwegen[2], Greet Vanden Berghe[1], and Dirk Van Oudheusden[2]

[1] KaHo Sint–Lieven
Information Technology
Gebroeders Desmetstraat 1, 9000 Gent, Belgium
{Wouter.Souffriau,Greet.VandenBerghe}@kahosl.be
[2] Katholieke Universiteit Leuven
Centre for Industrial Management
Celestijnenlaan 300A, 3001 Leuven (Heverlee), Belgium
{Pieter.Vansteenwegen,Dirk.Vanoudheusden}@cib.kuleuven.be

Summary. Developing metaheuristics requires in general a lot of work tuning different parameters. This paper presents a two–level algorithm to tackle this problem: an upper–level algorithm is used to determine the most appropriate set of parameters for a lower–level metaheuristic. This approach is applied to an Ant Colony Optimisation (ACO) metaheuristic that was designed to solve the Orienteering Problem (OP). That is a particular routing problem in which a score is earned for visiting a location. The objective is to maximise the sum of the scores, while not exceeding a given time budget. The ACO algorithm uses a set of ants that communicate through the environment by means of a pheromone trail. The transition rule and pheromone updating rules are influenced by a large number of parameters. These parameters are fine–tuned by a Genetic Algorithm (GA), which trains the ACO using test problems from the literature. The resulting ACO algorithm is compared with an exact algorithm by applying it to another set of problems. The scores obtained by the resulting algorithm are very near the optimal scores for the test problems.

Keywords: Orienteering problem, ant colony optimization, genetic algorithm, parameter tuning.

1 Introduction

The Orienteering Problem (OP) originated as a sport in which players get a map, a set of coordinates and a limited time budget. The purpose of the sport is to collect a maximum worth of prizes that are located at the coordinates, within the given time span. During the last twenty years a number of algorithms have been developed to tackle this problem, ranging from exact approaches over heuristics to metaheuristics. Feillet et al [10] present an overview of the best algorithms and more details will be discussed in Sect. 3.

This paper proposes a multi–level metaheuristic for solving the OP: an upper–level algorithm trains a lower–level algorithm that actually solves the problem.

C. Cotta et al. (Eds.): Adaptive and Multilevel Metaheuristics, SCI 136, pp. 255–269, 2008.
springerlink.com

In this paper, the lower–level uses an Ant Colony Optimisation (ACO) approach to tackle the OP. Obviously, this approach can be applied to other metaheuristics and other optimisation problems as well.

ACO techniques have been successfully applied to a number of combinatorial optimisation problems, such as the travelling salesman problem (TSP) [9], the quadratic assignment problem [11], the graph colouring problem [7] and the vehicle routing problem [4]. Metaheuristic approaches, like ACO, usually require a lot of work tuning the different parameters [9].

In nature, ants evolved and learned to cooperate in order to efficiently solve complex problems: they gather food, build nests, defend themselves from large predators, etc. We simulate this natural evolution using an upper–level genetic algorithm (GA), in order to train a general ACO algorithm in solving the OP. Instead of manually experimenting with numerous combinations of different transition rules, different pheromone updating rules and a wide variety of parameter settings, we let the GA decide on the internal fine–tuning of the ACO algorithm.

This paper is structured as follows: Sect. 2 formulates the OP as a mixed integer problem. Section 3 provides an overview of previous approaches, focussing on heuristic methods. Section 4 overviews the lower–level ACO approach. Section 5 explains how the upper–level GA is used in order to select well performing parameters from a huge parameter space. Section 6 compares the performance of the trained algorithm on a set of test problems from the literature. Finally, Sect. 7 concludes the paper.

2 The Orienteering Problem

The OP can be seen as a variant of the TSP. In the TSP, a collection of locations is given and the shortest route visiting them all has to be determined [8]. In the OP each location has a score and the total travelling time is limited. As a consequence not all locations can be visited. [13]. An important difference between the two is the objective function. The objective of the TSP is to limit the distance travelled, whereas in the OP the objective is to maximise the total score.

The standard OP can be formulated mathematically as follows: given are $n+1$ locations; the variable x_{ij} is equal to 1 if the solution contains an arc between the locations i and j, 0 otherwise; each location i has a score $S_i \geq 0$; location 0 is the starting location, location n is the end location; the shortest path between location i and location j requires time t_{ij}, the Euclidean distance between them; the total score has to be maximised, without exceeding a given time T_{max}.

$$Max \sum_{i=1}^{n-1} \sum_{j=1}^{n} S_i x_{ij} \tag{1}$$

$$\sum_{j=1}^{n-1} x_{0j} = \sum_{i=1}^{n-1} x_{in} = 1 \tag{2}$$

$$\sum_{i=0}^{n-1} x_{ik} = \sum_{j=1}^{n} x_{kj} \leq 1; \qquad \forall k = 1, \ldots, n-1 \tag{3}$$

$$1 \leq u_i \leq n; \qquad \forall i \neq 0 \tag{4}$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}); \qquad \forall i \neq 0, \forall j \neq 0 \tag{5}$$

$$\sum_{i=0}^{n-1} \sum_{j=1}^{n} t_{ij} x_{ij} \leq T_{max} \tag{6}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j = 0, \ldots, n \tag{7}$$

Constraint (2) states that a tour has to start at location 0 and has to end at location n. Constraint (3) states that no location should be visited more than once. Constraints (4) and (5) state that a single tour has to be constructed i.e. no subtours are allowed. These subtour elimination constraints use an extra variable u_i to order the locations in the tour, according to the MTZ formulation of the TSP [21]. Constraint (6) limits the available time i.e. the budget.

3 Previous Approaches to Solve the Orienteering Problem

This section provides a brief literature overview on the OP, concentrating on heuristic approaches. Ideas from these approaches will be used to develop the transition and pheromone updating rules of the ACO algorithm.

Tsiligirides [24] is the first to develop heuristic approaches for the OP. His first approach, based on a Monte Carlo technique, stochastically generates a large number of solutions. Paths are constructed using the following desirability measure for each location i: $A_i = (\frac{S_i}{t_{i,last}})^4$, where S_i is the score of location i and $t_{i,last}$ is the time from the current location of the tour to location i. A roulette wheel selection is used to select a location from the set of four locations with the largest desirability measures, which is then added to the current path.

Golden, Levy and Vohra [13] introduce a procedure that consists of three steps. In the first step a route is constructed, according to an insertion heuristic that assigns a weight to each location i: $W_i = a.S_i + b.T_i + c.E_i$, where T_i is the time from location i to the centre of gravity, E_i is the sum of the times from location i to the begin and end locations, and $a + b + c = 1$. In the second step, a 2–opt procedure [20] is performed and locations in the tour are exchanged with locations outside the tour in order to decrease the total tour length. In the third step, the centre of gravity is recomputed.

Golden, Wang and Liu [14] introduce a learning factor. They develop a new insertion heuristic, with the randomisation of Tsiligirides [24] and the centre of gravity concept [13]. The insertion heuristic now assigns weights to each location i, according to: $W_i = \alpha.S'_i + \beta.T'_i + \gamma.E'_i$, where S'_i is a score based on the score of

the location i, the scores of its neighbours, time to its neighbours and a learning component related to previous solutions, T_i' and E_i' are scaled versions of T_i and E_i, and $\alpha + \beta + \gamma = 1$.

Chao, Golden and Wasil [5] develop a method which does not focus on one tour, but keeps track of a number of feasible tours. First an ellipse is drawn, with the length of the main axis equal to the time budget of the problem and the starting and end location of the tour as foci. Only locations from this ellipse are considered in order to construct routes. The first step of the algorithm initialises a number of feasible tours. Next, the tour with the highest score is improved by a two–location exchange procedure, a one–location movement procedure and a clean up procedure. During this step the best tour may change. Finally, the best tour is saved, a reinitialisation procedure removes locations from that tour and the improvement step is repeated.

Liang and Smith [19] introduce an ACO approach, combined with a variable neighbourhood search. The following section will use the ideas from these heuristics to develop the transition rule of a standard ACO algorithm. All these heuristics have also been applied to the benchmark problem of Tsiligirides [24].

A recent literature overview on the OP can be found in [10].

4 Ant Colony Optimisation

ACO systems keep track of a number of possible tours through different ants, while using a learning component called pheromone to improve previous solutions [9]. The link with the learning component of Golden [14] and the method of Chao [5] is easy to make.

First the general ACO framework is discussed, then the solution components are discussed in more detail and finally a representation of the evaluation of the algorithm is presented.

4.1 Framework

Ants build solutions to the OP by moving probabilistically from one location to another, according to a transition rule, which decides which location the ant should go to next. The considered locations are in the candidate list. Initially the candidate list is calculated by drawing an ellipse, with the length of the main axis equal to the budget and the starting and end location of the tour as foci of the ellipse, as described in [5]. Ants choose their move probabilistically according to a transition function, which is based on local observations, such as the length of the arc to traverse and the amount of pheromone on it. After adding a location to the solution, the available budget is reduced by the time needed for the movement and a certain amount of pheromone evaporates. The evaporation of pheromone is also known as local pheromone update. At each step the locations that cannot be reached anymore within the current budget are removed from the candidate list. During an iteration of the algorithm k ants are placed in the k locations of the initial candidate list and each constructs a

tour. After each iteration, the best ant is allowed to drop pheromone on its path. This process is called global pheromone update. The number of iterations r is limited: when r is larger than a user–set maximum number, R_{max}, the algorithm stops. This paper uses $R_{max} = 20$.

4.2 Solution Components

This subsection discusses the solution components of the general framework in detail.

Initialise Pheromone Trail

Communication between different ants is performed through the environment, by means of a pheromone trail. Ants take the pheromone level of an arc into account when considering a transition to a next location. The amount of pheromone between location i and location j is given by $\tau_{i,j}$. When the algorithm starts, the pheromone trail is initialised using a number of components as follows:

$$\tau_0 = t_{init}^{x_1} s_{init}^{x_2} T_{max}^{x_3} n_{reach}^{x_4} t_{max}^{x_5} s_{max}^{x_6} \tag{8}$$

where t_{init} and s_{init} are respectively the time and the score of the tour of a single ant that traversed the problem with $\tau_0 = 1$; T_{max} is the available budget, n_{reach} is the number of locations that can be reached within the current budget, i.e. the number of locations contained in the ellipse; t_{max} is the longest time in the problem and s_{max} is the maximal score in the problem, out of which the unreachable locations are filtered. The behaviour of this function depends on the values of parameters $x_1 - x_6$.

Transition Rule

When an ant needs to choose a next location to visit, it uses the transition rule. The transition rule determines the probability for ant k to move from location i to location j. The transition probability is the product of a number of components. In order to construct the transition rule, a number of components can be used. The following components are based on the literature review of Sect. 3:

- the time–based component $t_{i,j}$ is the time of moving from location i to location j;
- the score–based component s_j is the score received for visiting location j;
- the centre–of–gravity–time–based component $t_{j,cog}$ is the time for moving from location j to the centre of gravity;
- the time–to–begin–and–end–location–based component $t_{begin,j,end} = t_{begin,j} + t_{j,end}$ is the time for moving from the starting location to location j plus the time for moving from location j to the end location.

Apart from components based on heuristics found in the literature, the following components are used:

- time–to–previous–location–based component $t_{prev,j}$ is the time for moving between the previously visited location $(i-1)$ to location j; The motivation for this component is that it could be better for the ants to make circular tours, i.e. moving as far as possible from the previously visited location.
- pheromone–based component $\tau_{i,j}$, the concentration of pheromone between location i and location j.

All these components are considered in the transition function T:

$$T_{i,j} = t_{i,j}^{x_7} s_j^{x_8} t_{j,cog}^{x_9} t_{begin,j,end}^{x_{10}} t_{prev,j}^{x_{11}} \tau_{i,j}^{x_{12}} \tag{9}$$

Parameters x_7 to x_{12} determine the interaction of the different components in the transition function. At each step, the ant calculates the transition function from its current location i to each possible location j, that does not belong to its current tour. When the transition values are known for the locations under consideration, the ant chooses its next move according to the following formula:

$$s = \begin{cases} ArgMax_{j \notin Tour}(T_{i,j}) & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases} \tag{10}$$

where q is a randomly chosen value with uniform probability in $[0,1]$ and q_0 $(0 \leq q_0 \leq 1)$ is a parameter. In case $q > q_0$ diversification is applied, by means of a roulette wheel: S is a random variable selected according to the following probability distribution [9]:

$$p_{ij} = \begin{cases} \frac{T_{i,j}}{\sum_{j \notin Tour} T_{i,j}} & \text{if } j \notin Tour \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Parameter q_0 influences the behaviour of the transition choice.

Local Pheromone Update

After an ant has made a transition from location i to location j, the pheromone level on this arc is updated. This is often called local updating and it is used in this case to avoid getting trapped in a local optimum. To avoid that all ants keep using the same arc, the pheromone level $\tau_{i,j}$ is brought towards its initial level τ_0 according to the following formula:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\tau_0 \tag{12}$$

Parameter ρ influences the behaviour of the local pheromone updating function.

Global Pheromone Update

After each iteration, when all ants have constructed a solution, the ant with the best tour is allowed to leave pheromone on its tour. This is called global pheromone updating, given by

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\Delta\tau_{i,j} \tag{13}$$

where $0 \leq \rho \leq 1$. The global updating should reflect the quality of the best solution. In order to calculate $\Delta\tau_{i,j}$, this paper uses the following formula:

$$\Delta\tau_{i,j} = t_{init}^{x_{13}} s_{init}^{x_{14}} n_{reach}^{x_{15}} t_{max}^{x_{16}} s_{max}^{x_{17}} t_{best}^{x_{18}} s_{best}^{x_{19}} \tag{14}$$

where t_{init} and s_{init} are respectively the time and the score of the tour of a single ant who traversed the problem with $\tau_0 = 1$; n_{reach} is the number of locations that can be reached within the current budget; t_{max} is the longest time in the problem and s_{max} is the maximal score in the problem; t_{best} and s_{best} are the time and the score of the ant that is allowed to release pheromone. The behaviour of this function depends on the values of the parameters x_{13} to x_{19}.

4.3 Solution Quality

When the ACO finishes after a number of iterations, it presents its best solution found for the problem at hand. The Solution Quality (SQ) for a given problem is defined as follows:

$$SQ = \frac{s_{best}}{s_{UB}} \tag{15}$$

where s_{best} is the score of the best solution found for the problem and s_{UB} is the theoretical upper bound for the score according to Leifer [17].

The SQ statistic will be used in the following section, to evaluate the ACO at hand.

5 Multi–level Structure

Based on the actual setting of the parameters, an ACO algorithm will perform differently. To determine the best set of parameters, an upper–level GA is used. This algorithm will simulate, evaluate and evolve different combinations of the parameters of the ACO algorithm, relieving the algorithm developer from the tedious task of manual parameterisation.

An upper–level GA is used to simulate the evolution of lower–level metaheuristics, i.e. ACO's. Each member of the population of the GA encodes solution components. These components are used in order to create an instance from a general ACO framework. A set of test problems is used to evaluate the instantiated ACO. The SQ is calculated after the run of the ACO on the test set. This SQ is used as an evaluation value for the chromosome. After each chromosome of the population of the GA is evaluated, selection, crossover and mutation are used to breed a new population of ACO algorithms.

Figure 1 illustrates the structure of the multi–level approach. The multi–level structure is used to train ants. Test sets 1 and 2 from Tsiligirides [24] are used as a training set. Test set 3 is used to verify that the resulting ACO algorithm is not over–fitted by means of the training set. First is explained how GAs work in general and the rest of this section explains how a GA is used to set the parameters of an ACO algorithm to solve the OP.

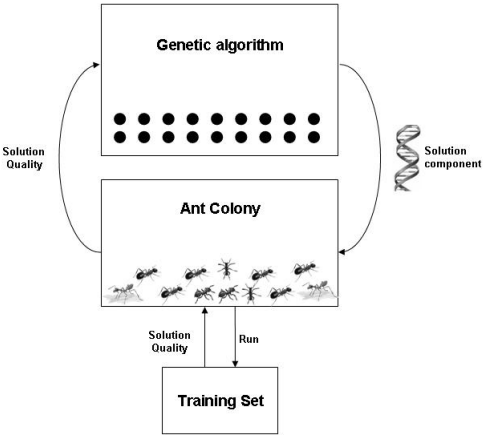


Fig. 1. Multi Level Structure

5.1 Genetic Algorithm

A GA is a search technique used to find approximate solutions for search and optimisation problems. A GA operates by iteratively updating a population of individuals, which are candidate solutions to the problem. The individuals are encoded as binary strings, called chromosomes, and are assigned a fitness value according to their evaluation. The algorithm starts by evaluating an initial, random population. Next the population is evolved by stochastically choosing parent individuals on which genetic operators are applied in order to recombine the parents and create new individuals, which are also evaluated. The process of selection, recombination and evaluation is called a generation or an epoch. As individuals with a higher fitness have a higher probability to be selected as a parent, the population evolves towards better solutions [12].

5.2 Chromosome Representation

Parameters $x_1, x_2, \ldots, x_{19}, q_0, \rho$ are to be optimised by the genetic algorithm. These parameters are encoded as real numbers. The domain of parameters $x_1, x_2, \ldots, x_{19}$ is $[-100, 100]$. The domain of parameters q_0 and ρ is $[0, 1]$. Each individual is encoded as a concatenation of these parameters and randomly initialised, e.g.

The motivation for this representation can be found in the building block hypothesis of Goldberg [12] which states "*instead of building high–performance*

Table 1. Example Chromosome Representation

	x_1	x_2	x_3	x_4	x_5 ...	x_{18}	x_{19}	q_0	ρ
Individual 1	31.42	97.08	−21.05	36.34	−76.59 ...	34.93	−59.26	0.54	0.38
Individual 2	22.94	−63.92	−54.18	−37.93	8.32 ...	37.92	52.49	0.95	0.04

strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings". By using a concatenation of real numbers, instead of a concatenation of bits, individuals are not disrupted by genetic operators at illogical positions, which would otherwise lead to a waste of valuable computational resources.

5.3 Selection and Population Replacement

During each generation, a set of the existing individuals is selected as parents and is allowed to produce offspring. In this paper we use the stochastic universal selection algorithm, according to Baker [3]. In this selection method, the probability of selecting an individual is based on its fitness, like in roulette wheel selection. Reeves [23] compares this method to an "*equally spaced multi–armed spinner*".

This paper uses a steady–state population replacement approach: the top 2% of the old population automatically migrates to the next generation. A population size of 100 is used.

5.4 Genetic Operators

The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population [22]. A uniform crossover operator is used with a probability $P_c = 0.9$. This operator randomly generates a crossover mask: a binary string with the same length as the chromosomes. The value of the bits in the crossover mask determine how the chromosome information of the parents is interchanged in order to create two offsprings. If the value of a bit is 1, chromosome information is copied from the first parent, if the value is 0, information from the second parent is used and vice–versa for the second offspring, as illustrated here:

Table 2. Example Uniform Crossover Operator

	x_1	x_2	x_3	x_4	x_5 ...	x_{18}	x_{19}	q_0	ρ
Individual 1	31.42	97.08	−21.05	36.34	−76.59 ...	34.93	−59.26	0.54	0.38
Individual 2	22.94	−63.92	−54.18	−37.93	8.32 ...	37.92	52.49	0.95	0.04
Crossover mask	1	0	0	0	1 ...	0	1	1	1
Offspring 1	31.42	−63.92	−54.18	−37.93	−76.59 ...	37.92	−59.26	0.54	0.38
Offspring 2	22.94	97.08	−21.05	36.34	8.32 ...	34.93	52.49	0.95	0.04

The mutation operator is applied with a probability $P_m = 0.001$ and replaces the value of the parameter under consideration with a value drawn uniformly from its domain.

5.5 Evaluation

Each individual is evaluated against test set 1 and 2 from Tsiligirides [24] which consists of a total of 29 OP instances. The third test set, containing 20 instances, will be used later as an unseen test, for validation purposes, in order to examine the generality of the learned solution.

The Solution Quality described in Sect. 4.3 is the base for calculating the evaluation value of a chromosome. The evaluation function averages the SQ over the evaluated n problem instances:

$$E = 100\frac{\sum_{i=1\rightarrow n} SQ_i}{n} \tag{16}$$

where i is the index of the test problem, n is the total number of test problems, SQ_i is the Solution Quality for test problem i. This evaluation value will be used by the GA to update and improve the population.

6 Computational Experience

The lower–level ACO is implemented in J2SDK 1.5.0 from Sun [1]. The upper–level GA is implemented in C, by means of the Parallel Genetic Algorithm Library [18]. The communication between the two levels is carried out by exchanging temporary files. The experiment is run on a linux cluster, interconnected by an Infiniband network. The experiment is allowed to run four hours on ten computational nodes, for a total of forty hours.

The experiment tries to optimise the SQ of test sets 1 and 2 from Tsiligirides [24]. Figure 2 illustrates a single run of the GA: for each generation, the fitness of the best and the worst individual are shown, and also the average fitness of the total population.

In the first generation, the best parameter combination has been found with a $SQ = 81.99$. In generation 163, the best parameter combination has been found with a $SQ = 93.56$. The relatively good result in the first generation can probably be explained by stating that the ratio between the parameters is important, next to the absolute value of the parameters. When initialising the population, a relatively good ratio is randomly generated.

Table 3 contains the values of the parameters of the best chromosome. x_{11}, the parameter which influences $t_{prev,j}$ in the transition function, is equal to 20.14417. This gives an indication that it is useful to take the distance to the previous location into account when considering a probabilistic transition to a next location. As the value is positive, locations which are further away from the previous location are more likely to be visited. x_{15}, the parameter which influences n_{reach} in the calculation of $\Delta\tau_{i,j}$, is nearly equal to 0. It leads to the conclusion that it would be better not to take n_{reach} into account.

Tables 4 to 6 compare three instances of the ACO algorithm: $ACO^{untrained}_{R_{max}=20}$ is not trained by the GA and is allowed to run for 20 iterations; $ACO^{untrained}_{R_{max}=40}$ is not trained by the GA and is allowed to run for 40 iterations; $ACO^{trained}_{R_{max}=20}$

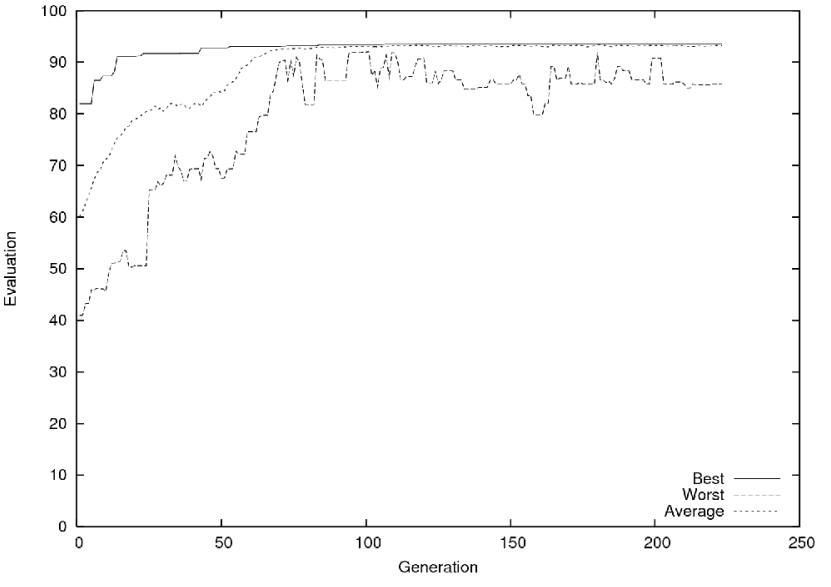


Fig. 2. GA Evolution

Table 3. Parameter values from best chromosome

Param.	Value	Param.	Value	Param.	Value	Param.	Value
x_1	-41.07874	x_7	-56.25024	x_{12}	14.03927	x_{17}	63.44217
x_2	-38.11543	x_8	26.87057	x_{13}	60.7474	x_{18}	69.67535
x_3	69.59213	x_9	17.36928	x_{14}	95.1037	x_{19}	84.78181
x_4	-85.22636	x_{10}	-1.676184	x_{15}	-0.073423	q_0	0.2356631
x_5	24.03524	x_{11}	20.14417	x_{16}	83.67101	ρ	0.6423123
x_6	87.27342						

is trained by GA and is allowed to run for 20 iterations. The scores of the three algorithms are compared with the optimal solutions on all the test sets from Tsiligirides [24], including the third test set, which was not used for training. The untrained ACOs use parameter settings from the best chromosome of the initial population of the GA, while the trained ACO uses parameter settings of the best chromosome of the evolved GA. ILOG CPLEX is used to solve the model introduced in Sect. 2 to optimality. The last line of the table contains the percentage comparison between the ACOs and the optimal solutions. It is interesting to note that the application of the trained ACO to the unseen test set 3, scores equally high (98.21%) as test set 1 (98.20%) and better than test set 2 (95.22%). For the total 49 test problems, the results of the trained ACO algorithm differ on average only 2.47% from the optimal score. It can be concluded that the parameter settings of the best individual can now be used to solve other instances of the OP. The results of untrained ACO ($ACO^{untrained}_{R_{max}=20}$),

Table 4. Comparison on Tsiligirides' test set 1

Budget	Optimal	$ACO^{untrained}_{R_{max}=20}$	$ACO^{untrained}_{R_{max}=40}$	$ACO^{trained}_{R_{max}=20}$
5	10	10	10	10
10	15	15	15	15
15	45	35	35	45
20	65	45	45	65
25	90	55	65	90
30	110	85	85	110
35	135	110	110	130
40	155	125	125	150
46	175	150	150	165
50	190	155	155	180
55	205	175	175	195
60	225	190	190	220
65	240	205	205	240
70	260	220	220	255
73	265	225	225	260
75	270	235	235	265
80	280	250	250	275
85	285	255	255	285
%Optimal		83.63%	84.25%	98.20%

Table 5. Comparison on Tsiligirides' test set 2

Budget	Optimal	$ACO^{untrained}_{R_{max}=20}$	$ACO^{untrained}_{R_{max}=40}$	$ACO^{trained}_{R_{max}=20}$
15	120	90	90	120
20	200	140	140	200
23	210	180	180	200
25	230	200	200	210
27	230	200	200	230
30	265	205	205	260
32	300	230	230	260
35	320	230	230	305
38	360	250	250	350
40	395	265	285	365
45	450	300	340	410
%Optimal		75.79%	77.06%	95.22%

which was allowed to run for an equal number of iterations, differ on average 18.15% from the optimal score. This illustrates the usefulness of the training by the parallel GA. The average runtime of the untrained ACO allowed to run for 20 iterations ($ACO^{untrained}_{R_{max}=20}$) is 720 ms and its results differ on average 18.15% from the optimal score, while the average runtime of the untrained ACO allowed to run for 40 iterations ($ACO^{untrained}_{R_{max}=40}$) is 989 ms and its results differ on average 17.19% from the optimal score. It can be concluded that parameter learning across short runs of the low–level heuristic is worthwile, as increasing the runtime

Table 6. Comparison on Tsiligirides' test set 3

Budget	Optimal	$ACO^{untrained}_{R_{max}=20}$	$ACO^{untrained}_{R_{max}=40}$	$ACO^{trained}_{R_{max}=20}$
15	170	110	120	170
20	200	160	160	180
25	260	190	190	260
30	320	230	230	320
35	390	280	280	380
40	430	360	360	430
45	470	390	430	460
50	520	440	440	520
55	550	500	500	550
60	580	520	520	560
65	610	570	570	600
70	640	570	570	630
75	670	610	610	660
80	710	640	640	700
85	740	640	640	720
90	770	650	670	750
95	790	660	700	750
100	800	670	670	790
105	800	700	700	800
110	800	710	710	800
%Optimal		83.57%	84.68%	98.21%

of the stand–alone untrained ACO does not significantly improve the quality of the results in comparison with the results of the trained ACO.

To check the sensitivity of the approach, three GA runs were performed, which gave similar results. As expected, only one run would have been enough to determine the parameters of the lower–level metaheuristic.

The results published by Chao [5] are equal to the optimal solution. The ACO alone is not capable of finding these optimal solutions for every test problem. A local search method on top of the best ant of each iteration can improve the solution of the ACO algorithm [19].

7 Conclusions and Future Research

The contribution of this paper is the introduction of a metaheuristic approach that deals with fine–tuning its own parameters and that successfully solves test instances of the OP almost to optimality (98%). The metaheuristic approach consists of a lower–level ACO and a upper–level GA. No preliminary testing to determine a good value for a series of parameters is necessary.

The promising results of the experiments open up opportunities for further research, i.e. more advanced GA techniques and multi–objective optimisation.

The training could be improved by applying more advanced GA techniques, from the field of continuous optimisation. Specialised genetic operators for real–coded GAs should be considered in order to increase the efficiency of the search [6, 15, 16]. Also, advanced parallel approaches could be considered in order to guard the balance between exploration and exploitation of the parameter space during the training [2, 15] and to avoid wasting CPU resources by intelligently reducing the size of the population [6].

The training approach could be extended by using a multi–objective approach to design the lower–level heuristic, in order to take multiple conflicting objectives into account, such as speed of the algorithm and its simplicity. The speed can be denoted by the number of iterations required to find the best solution. As simplicity, parameter values closer to zero can be given a higher evaluation value. The radix of a parameter value close to zero can be omitted from the algorithm.

GAs appear to be very useful for parameter selection and automated tuning of different kinds of metaheuristics with parameters. The results of the paper suggest that applying the multi–level approach to different optimisation problems is very likely to be successful.

Acknowledgements

The authors gratefully acknowledge Argonne's Center for Computational Science and Technology for providing the PGAPack Parallel Genetic Algorithm Library [18].

References

1. http://java.sun.com/j2se/1.5.0/
2. Alba, E., Luna, F., Nebro, A.: Advances in parallel heterogeneous genetic algorithms for continuous optimization. International Journal of Applied Mathematics and Computer Science 14(3), 317–333 (2004)
3. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the 2nd International Conference on Genetic Algorithms, Hillsdale, New Jersey, pp. 14–21. Lawrence Erlbaum Associates, Mahwah (1987)
4. Bullnheimer, B.: Ant Colony Optimization in Vehicle Routing. PhD thesis, University of Vienna (1999)
5. Chao, I.-M., Golden, B., Wasil, E.: A fast and effective heuristic for the orienteering problem. European Journal of Operational Research 88(3), 475–489 (1996)
6. Chelouah, R., Siarry, P.: A continuous genetic algorithm designed for the global optimization of multimodal functions. Journal of Heuristics 6, 191–213 (2000)
7. Costa, D., Hertz, A.: Ants can colour graphs. J. Oper. Res. Soc. 48, 295–305 (1997)
8. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling salesman problem. Operations Research 2, 393–410 (1954)
9. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comp. 1, 53–66 (1997)
10. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Transportation Science 39, 188–205 (2005)

11. Gambardella, L.M., Taillard, E.D., Dorigo, M.: Ant colonies for the qap. J. Oper. Res. Soc. 50(2), 167–176 (1999)
12. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Kluwer Academic Publishers, Dordrecht (1989)
13. Golden, B., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics 34, 307–318 (1987)
14. Golden, B., Wang, Q., Liu, L.: A multifaceted heuristic for the orienteering problem. Naval Research Logistics 35, 359–366 (1988)
15. Herrera, F., Lozano, M.: Gradual distributed real-coded genetic algorithms. IEEE Transactions on Evolutionary Computation 4(1), 43–63 (2000)
16. Herrera, F., Lozano, M., Verdegay, J.: Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. Artificial Intelligence Review 12(4), 265–319 (1998)
17. Leifer, A.C., Rosenwein, M.S.: Strong linear programming relaxations for the orienteering problem. Bell System Technical Journal 44, 2245–2269 (1994)
18. Levine, D.: Parallel genetic algorithm library, http://www-fp.mcs.anl.gov/CCST/research/reports_pre1998/comp_bio/stalk/pgapack.html
19. Liang, Y.-C., Smith, A.E.: An ant colony approach to the orienteering problem. Technical report, Department of Industrial and Systems Engineering, Auburn University, Auburn, AL (2001)
20. Lin, S.: Computer solutions of the traveling salesman problem. Bell System Technical Journal 44, 2245–2269 (1965)
21. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulations and traveling salesman problems. J. ACM 7, 326–329 (1960)
22. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
23. Reeves, C.: Handbook of Metaheuristics, chapter Genetic Algorithms, pp. 55–82. Kluwer Academic Publishers, Dordrecht (2003)
24. Tsiligirides, T.: Heuristic methods applied to orienteering. J. Oper. Res. Soc. 35(9), 797–809 (1984)

Index

Author Index