

**Thomas Bartz-Beielstein
Jürgen Branke
Bogdan Filipič
Jim Smith (Eds.)**

LNCS 8672

Parallel Problem Solving from Nature - PPSN XIII

**13th International Conference
Ljubljana, Slovenia, September 13–17, 2014
Proceedings**



 **Springer**

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Thomas Bartz-Beielstein Jürgen Branke
Bogdan Filipič Jim Smith (Eds.)

Parallel Problem Solving from Nature – PPSN XIII

13th International Conference
Ljubljana, Slovenia, September 13-17, 2014
Proceedings



Springer

Volume Editors

Thomas Bartz-Beielstein
Cologne University of Applied Sciences
Faculty of Computer and Engineering Sciences
Steinmüllerallee 1, 51643 Gummersbach, Germany
E-mail: thomas.bartz-beielstein@fh-koeln.de

Jürgen Branke
University of Warwick, Warwick Business School
Coventry, CV8 2SY, UK
E-mail: juergen.branke@wbs.ac.uk

Bogdan Filipič
Jožef Stefan Institute, Department of Intelligent Systems
Jamova cesta 39, 1000 Ljubljana, Slovenia
E-mail: bogdan.filipic@ijs.si

Jim Smith
University of the West of England
Department of Computer Science and Creative Technologies
Bristol, BS16 1QY, UK
E-mail: james.smith@uwe.ac.uk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-10761-5 e-ISBN 978-3-319-10762-2
DOI 10.1007/978-3-319-10762-2
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014946588

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

© Springer International Publishing Switzerland 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This LNCS volume contains the proceedings of the 13th International Conference on Parallel Problem Solving from Nature (PPSN 2014). This biennial event constitutes one of the most important and highly regarded international conferences in evolutionary computation and bio-inspired metaheuristics. Continuing with a tradition that started in Dortmund in 1990, PPSN 2014 was held during September 13–17, 2014, in Ljubljana, Slovenia.

PPSN 2014 received 217 submissions from 44 countries. After an extensive peer-review process where most papers were evaluated by at least four reviewers, the Program Committee chairs went through all the reports and ranked the papers. The top 90 manuscripts were finally selected for inclusion in this LNCS volume and for presentation at the conference. This represents an acceptance rate of 41%, which guarantees that PPSN will continue to be one of the most respected conferences for researchers working in natural computing around the world.

PPSN 2014 featured three distinguished keynote speakers: Jadran Lenarčič (Jožef Stefan Institute, Slovenia), Thomas Bäck (Leiden University, The Netherlands), and A. E. (Gusz) Eiben (VU University Amsterdam, The Netherlands).

The meeting began with seven workshops: “Scaling Behaviors of Landscapes, Parameters and Algorithms” (Ender Özcan, Andrew J. Parkes), “Advances in Multimodal Optimization” (Mike Preuss, Michael G. Epitropakis, Xiaodong Li), “Semantic Methods in Genetic Programming” (Colin Johnson, Krzysztof Krawiec, Alberto Moraglio, Michael O’Neill), “In Search of Synergies Between Reinforcement Learning and Evolutionary Computation” (Madalina M. Drugan, Bernard Manderick), “Natural Computing for Protein Structure Prediction” (José Santos Reyes, Gregorio Toscano, Julia Handl), “Workshop on Nature-Inspired Techniques for Robotics” (Claudio Rossi, Nicolas Bredeche, Kasper Stoy), and “Student Workshop on Bioinspired Optimization Methods and their Applications – BIOMA 2014” (Jurij Šilc, Aleš Zamuda). The workshops offered an ideal opportunity for the conference members to explore specific topics in evolutionary computation, bio-inspired computing, and metaheuristics in an informal and friendly setting.

PPSN 2014 also included nine free tutorials: “Theory of Evolutionary Computation” (Anne Auger, Benjamin Doerr), “Low or No Cost Distributed Evolutionary Computation” (J.J. Merelo), “Cartesian Genetic Programming” (Julian F. Miller), “Multimodal Optimization” (Mike Preuss), “Theory of Parallel Evolutionary Algorithms” (Dirk Sudholt), “Automatic Design of Algorithms via Hyper-heuristic Genetic Programming” (John R. Woodward, Jerry Swan, Michael Epitropakis), “Evolutionary Bilevel Optimization” (Ankur Sinha, Pekka Malo, Kalyanmoy Deb), “Parallel Experiences in Solving Complex Problems”

(Enrique Alba), and “Algorithm and Experiment Design with HeuristicLab – An Open Source Optimization Environment for Research and Education” (Stefan Wagner, Gabriel Kronberger).

We wish to express our gratitude in particular to the Program Committee members and external reviewers who provided thorough evaluations of all 217 submissions. We also express our profound thanks to all the members of the Organizing Committee and the local organizers for their outstanding efforts in preparing for and running the conference. Thanks to all the keynote and tutorial speakers for their participation, which greatly enhanced the quality of the conference. Finally, we also express our gratitude to the sponsoring institutions for their financial support and the conference partners for participating in the organization of this event.

September 2014

Thomas Bartz-Beielstein
Jürgen Branke
Bogdan Filipič
Jim Smith

Local Organizing Committee

Vesna Koricki-Špetič	Jožef Stefan Institute, Slovenia
Peter Korošec	Jožef Stefan Institute, Slovenia
Blaž Mahnič	Jožef Stefan Institute, Slovenia
Jurij Šilc	Jožef Stefan Institute, Slovenia
Vida Vukašinić	Jožef Stefan Institute, Slovenia

Steering Committee

Carlos Cotta	Universidad de Málaga, Spain
David W. Corne	Heriot-Watt University Edinburgh, UK
Kenneth De Jong	George Mason University, USA
Agoston E. Eiben	VU University Amsterdam, The Netherlands
Juan Julián Merelo Guervós	Universidad de Granada, Spain
Günter Rudolph	Dortmund University of Technology, Germany
Thomas P. Runarsson	University of Iceland, Iceland
Robert Schaefer	University of Krakow, Poland
Marc Schoenauer	Inria, France
Xin Yao	University of Birmingham, UK

Workshops

Scaling Behaviors of Landscapes, Parameters and Algorithms

Ender Özcan and Andrew J. Parkes

Advances in Multimodal Optimization

Mike Preuss, Michael G. Epitropakis, and Xiaodong Li

Semantic Methods in Genetic Programming

Colin Johnson, Krzysztof Krawiec, Alberto Moraglio, and Michael O'Neill

In Search of Synergies Between Reinforcement Learning and Evolutionary Computation

Madalina M. Drugan and Bernard Manderick

Natural Computing for Protein Structure Prediction

José Santos Reyes, Gregorio Toscano, and Julia Handl

Workshop on Nature-Inspired Techniques for Robotics

Claudio Rossi, Nicolas Bredeche, and Kasper Stoy

Student Workshop on Bioinspired Optimization Methods and Their Applications – BIOMA 2014

Jurij Šilc and Aleš Zamuda

Tutorials

Theory of Evolutionary Computation

Anne Auger and Benjamin Doerr

Multimodal Optimization

Mike Preuss

Evolutionary Bilevel Optimization

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb

Low or No Cost Distributed Evolutionary Computation

J.J. Merelo

Theory of Parallel Evolutionary Algorithms

Dirk Sudholt

Parallel Experiences in Solving Complex Problems

Enrique Alba

Cartesian Genetic Programming

Julian F. Miller

Automatic Design of Algorithms via Hyper-heuristic Genetic Programming

John R. Woodward, Jerry Swan, and Michael Epitropakis

Algorithm and Experiment Design with HeuristicLab – An Open Source Optimization Environment for Research and Education

Stefan Wagner and Gabriel Kronberger

Keynote Speakers

Jadran Lenarčič, Jožef Stefan Institute, Slovenia

Thomas Bäck, Leiden University, The Netherlands

Agoston E. Eiben, VU University Amsterdam, The Netherlands

Sponsoring Institutions

B2 d.o.o.

Flaška d.d.

Kolektor Group d.o.o.

Conference Partners

City of Ljubljana

GR – Ljubljana Exhibition and Convention Centre

Slovenian Artificial Intelligence Society
Toleranca marketing d.o.o.

Program Committee

Youhei Akimoto	Shinshu University, Japan
Enrique Alba	University of Málaga, Spain
Dirk Arnold	Dalhousie University, Canada
Gideon Avigad	Braude College, Israel
Dogan Aydin	Dumlupinar University, Turkey
Jaume Bacardit	University of Nottingham, UK
Helio Barbosa	Laboratório Nacional de Computação Científica, Brazil
Thomas Bartz-Beielstein	Cologne University of Applied Sciences, Germany
Simone Bassis	University of Milan, Italy
Roberto Battiti	Università di Trento, Italy
Gerardo Beni	Bourns College of Engineering, USA
Heder Bernardino	Universidade Federal de Juiz de Fora, Brazil
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Mauro Birattari	IRIDIA, Université Libre de Bruxelles, Belgium
Christian Blum	University of Basque Country, Spain
Yossi Borenstein	VisualDNA, UK
Borko Bošković	University of Maribor, Slovenia
Peter Bosman	Centrum Wiskunde & Informatica (CWI), The Netherlands
Pascal Bouvry	University of Luxembourg
Anthony Brabazon	University College Dublin, Ireland
Jürgen Branke	University of Warwick, UK
Dimo Brockhoff	Inria Lille - Nord Europe, France
Will Browne	Victoria University of Wellington, New Zealand
Larry Bull	University of the West of England, UK
Tadeusz Burczynski	Silesian University of Technology, Poland
Edmund Burke	University of Stirling, UK
Stefano Cagnoni	University of Parma, Italy
Ying-Ping Chen	National Chiao Tung University, Taiwan
Miroslav Chlebik	University of Sussex, UK
Sung-Bae Cho	Yonsei University, South Korea
Siang Yew Chong	University of Nottingham, Malaysia
Carlos Coello Coello	CINVESTAV-IPN, México
David Corne	Heriot-Watt University, UK
Ernesto Costa	University of Coimbra, Portugal
Jole Costanza	University of Catania, Italy
Carlos Cotta	University of Malaga, Spain

Peter Cowling	University of York, UK
Kenneth De Jong	George Mason University, USA
Antonio Della Cioppa	Natural Computation Lab, DIIE, University of Salerno, Italy
Luca Di Gaspero	DIEGM - University of Udine, Italy
Benjamin Doerr	Max Planck Institute for Informatics, Germany
Carola Doerr	Max Planck Institute for Informatics, Germany
Marco Dorigo	Université Libre de Bruxelles, Belgium
Rafal Drezewski	AGH University of Science and Technology, Poland
Jeremie Dubois-Lacoste	Université Libre de Bruxelles, Belgium
Gusz Eiben	Vrije Universiteit Amsterdam, The Netherlands
Aniko Ekart	Aston University, UK
Talbi El-Ghazali	University of Lille, France
Michael Emmerich	LIACS, Leiden University, The Netherlands
Margaret Eppstein	University of Vermont, USA
Anton Eremeev	Omsk Branch of Sobolev Institute of Mathematics, SB RAS, Russia
Bogdan Filipič	Jožef Stefan Institute, Slovenia
Steffen Finck	Vorarlberg University of Applied Sciences, Austria
Andreas Fischbach	Cologne University of Applied Sciences, Germany
Iztok Fister	University of Maribor, Slovenia
Oliver Flasch	Cologne University of Applied Sciences, Germany
Carlos M. Fonseca	University of Coimbra, Portugal
Tobias Friedrich	Friedrich-Schiller-Universität Jena, Germany
Martina Frieze	Cologne University of Applied Sciences, Germany
Marcus Gallagher	University of Queensland, Australia
Jonathan M Garibaldi	University of Nottingham, UK
Mario Giacobini	University of Turin, Italy
Tobias Glasmachers	RUB, Germany
Roderich Gross	Sheffield University, UK
Steven Gustafson	GE Global Research, UK
Walter Gutjahr	University of Vienna, Austria
Pauline Haddow	Norwegian University of Science and Technology, Norway
Hisashi Handa	Kindai University, Japan
Julia Handl	University of Manchester, UK
Nikolaus Hansen	Inria Saclay, France
Jin-Kao Hao	University of Angers, France
Emma Hart	Napier University, UK

Verena Heidrich-Meisner	Extraterrestrial Physics, CAU Kiel, Germany
Torsten Hildebrandt	Universität Bremen, Germany
Christian Igel	Institut für Neuroinformatik, Germany
Pedro Isasi Viñuela	Carlos III University of Madrid, Spain
Hisao Ishibuchi	Osaka Prefecture University, Japan
Christian Jacob	University of Calgary, Canada
Thomas Jansen	Aberystwyth University, UK
Bryant Julstrom	St. Cloud State University, UK
George Karakostas	McMaster University, Canada
Andy Keane	University of Southampton, UK
Graham Kendall	University of Nottingham, UK
Joshua Knowles	University of Manchester, UK
Barbara Koroušić Seljak	Jožef Stefan Institute, Slovenia
Krzysztof Krawiec	Poznan University of Technology, Poland
Halina Kwasnicka	Wroclaw University of Technology, Poland
Timo Kötzing	Max Planck Institute for Informatics, Germany
Joerg Laessig	University of Applied Sciences Zittau/Görlitz, Germany
Dario Landa-Silva	University of Nottingham, UK
Pier Luca Lanzi	Politecnico di Milano, Italy
Per Kristian Lehre	University of Nottingham, UK
Peter Lewis	Aston University, UK
Xiaodong Li	RMIT University, Australia
Tianjun Liao	Université Libre de Bruxelles, Belgium
Giosue' Lo Bosco	Università di Palermo, Italy
Fernando Lobo	University of Algarve, Portugal
Daniele Loiacono	Politecnico di Milano, Italy
Jose A. Lozano	The University of the Basque Country, Spain
Simon Lucas	University of Essex, UK
Evelyne Lutton	Inria, France
Manuel López-Ibáñez	Université Libre de Bruxelles, Belgium
Vittorio Maniezzo	University of Bologna, Italy
Elena Marchiori	Radboud University, The Netherlands
Carlos Martin-Vide	Rovira i Virgili University, Spain
Benedetto Matarazzo	University of Catania, Italy
Giancarlo Mauri	University of Milano-Bicocca, Italy
Jacek Mańdziuk	Warsaw University of Technology, Poland
Alexander Melkozerov	Tomsk State University of Control Systems and Radioelectronics, Russia
J.J. Merelo	Universidad de Granada, Spain
Marjan Mernik	University of Maribor, Slovenia
Silja Meyer-Nieberg	Universität der Bundeswehr München, Germany
Martin Middendorf	University of Leipzig, Germany
Kaisa Miettinen	University of Jyväskylä, Finland
Julian Miller	University of York, UK

Edmondo Minisci	University of Strathclyde, UK
Marco Montes de Oca	University of Delaware, USA
Sipper Moshe	Ben-Gurion University, Israel
Sanaz Mostaghim	Institute AIFB, Germany
Boris Naujoks	Cologne University of Applied Sciences, Germany
Ferrante Neri	De Montfort University, UK
Frank Neumann	The University of Adelaide, Australia
Michael O'Neill	University College Dublin, Ireland
Gabriela Ochoa	University of Stirling, UK
Pietro Oliveto	The University of Sheffield, UK
Yew-Soon Ong	Nanyang Technological University, Singapore
Gregor Papa	Jožef Stefan Institute, Slovenia
Gisele Pappa	UFMG, Brazil
Luis Paquete	CISUC, University of Coimbra, Portugal
Andrew J. Parkes	University of Nottingham, UK
Ian Parmee	Advanced Computational Technologies, UK
Marco Pavone	Stanford University, USA
Martin Pelikan	Google, USA
David Pelta	University of Granada, Spain
Silvia Poles	EnginSoft, Belgium
Petr Pošík	Czech Technical University in Prague, Czech Republic
Richard Preen	University of the West of England, UK
Mike Preuss	TU Dortmund University, Germany
William Rand	University of Maryland, USA
Khaled Rasheed	University of Georgia, USA
Tapabrata Ray	University of New South Wales, Australian Defence Force Academy, Australia
Eduardo Rodriguez-Tello	CINVESTAV-Tamaulipas, Mexico
Andrea Roli	Alma Mater Studiorum - Università di Bologna, Italy
Günter Rudolph	TU Dortmund University, Germany
Thomas Runarsson	University of Iceland
Thomas A. Runkler	Siemens Corporate Technology, Germany
Conor Ryan	University of Limerick, Ireland
Erol Sahin	Middle East Technical University, Turkey
Ivo Sbalzarini	Max Planck Institute of Molecular Cell Biology and Genetics, Germany
Robert Schaefer	AGH University of Science and Technology, Poland
Marc Schoenauer	Inria Saclay, Orsay Cedex, France
Oliver Schuetze	CINVESTAV-IPN, Mexico
Michèle Sebag	University of Paris-Sud, CNRS, France
Martin Serpell	University of the West of England, UK
Roberto Serra	University of Modena and Reggio Emilia, Italy

Marc Sevaux	Lab-STICC, Université de Bretagne-Sud, France
Jonathan Shapiro	University of Manchester, UK
Christopher Simons	University of the West of England, UK
Jim Smith	University of the West of England, UK
Christine Solnon	LIRIS CNRS UMR 5205/INSA Lyon, France
Terence Soule	University of Idaho, USA
Catalin Stoean	University of Craiova, Romania
Jörg Stork	Cologne University of Applied Sciences, Germany
Thomas Stuetzle	Université Libre de Bruxelles, Belgium
Dirk Sudholt	University of Sheffield, UK
Ponnuthurai Suganthan	Nanyang Technological University, Singapore
Jerry Swan	University of Stirling, UK
Kay Chen Tan	National University of Singapore
Daniel Tauritz	Missouri University of Science and Technology, USA
Jorge Tavares	University of Coimbra, Portugal
German Terrazas Angulo	University of Nottingham, UK
Andrea Tettamanzi	Université de Nice Sophia Antipolis, France
Lothar Thiele	ETH Zurich, Switzerland
Dirk Thierens	Universiteit Utrecht, The Netherlands
Jon Timmis	University of York, UK
Jerzy Tiuryn	Warsaw University, Poland
Heike Trautmann	TU Dortmund University, Germany
Elio Tuci	Aberystwyth University, UK
Tea Tušar	Jožef Stefan Institute, Slovenia
Rasmus Ursem	Grundfos Research Technology, Denmark
Leonardo Vanneschi	NOVA School of Statistics and Information Management, Portugal
Sébastien Verel	Inria Lille Nord Europe, France
Markus Wagner	School of Computer Science, Australia
Lipo Wang	Nanyang Technological University, Singapore
Man Leung Wong	Lingnan University, Hong Kong, SAR China
Ning Xiong	Mälardalen University, Sweden
Shengxiang Yang	De Montfort University, UK
Gary Yen	Oklahoma State University, USA
Tina Yu	Memorial University of Newfoundland, Canada
Yang Yu	Nanjing University, China
Martin Zaefferer	Cologne University of Applied Sciences, Germany
Aleš Zamuda	UM FERI, Slovenia
Christine Zarges	University of Birmingham, UK
Qingfu Zhang	University of Essex, UK

Table of Contents

Keynote Papers

Some Computational Aspects of Robot Kinematic Redundancy	1
<i>Jadran Lenarčič</i>	
Power Distribution Network Reconfiguration by Evolutionary Integer Programming	11
<i>Kaifeng Yang, Michael T.M. Emmerich, Rui Li, Ji Wang, and Thomas Bäck</i>	
In Vivo Veritas: Towards the Evolution of Things	24
<i>Agoston Endre Eiben</i>	

Adaptation, Self-Adaptation and Parameter Tuning

Online Black-Box Algorithm Portfolios for Continuous Optimization . . .	40
<i>Petr Baudiš and Petr Pošík</i>	
Shuffle and Mate: A Dynamic Model for Spatially Structured Evolutionary Algorithms	50
<i>Carlos M. Fernandes, Juan L.J. Laredo, Juan Julian Merelo, Carlos Cotta, Rafael Nogueras, and Agostinho C. Rosa</i>	
How to Assess Step-Size Adaptation Mechanisms in Randomised Search	60
<i>Nikolaus Hansen, Asma Atamna, and Anne Auger</i>	
Maximum Likelihood-Based Online Adaptation of Hyper-Parameters in CMA-ES	70
<i>Ilya Loshchilov, Marc Schoenauer, Michèle Sebag, and Nikolaus Hansen</i>	
Run-Time Parameter Selection and Tuning for Energy Optimization Algorithms	80
<i>Ingo Mauser, Marita Dorscheid, and Hartmut Schmeck</i>	
Towards a Method for Automatic Algorithm Configuration: A Design Evaluation Using Tuners	90
<i>Elizabeth Montero and María-Cristina Riff</i>	
Parameter Prediction Based on Features of Evolved Instances for Ant Colony Optimization and the Traveling Salesperson Problem	100
<i>Samadhi Nallaperuma, Markus Wagner, and Frank Neumann</i>	

Self-Adaptive Genotype-Phenotype Maps:
 Neural Networks as a Meta-Representation 110
Luís F. Simões, Dario Izzo, Evert Haasdijk, and Agoston Endre Eiben

The Baldwin Effect Hinders Self-Adaptation 120
Jim Smith

On Low Complexity Acceleration Techniques for Randomized
 Optimization 130
Sebastian Urban Stich

Stopping Criteria for Multimodal Optimization 141
Simon Wessing, Mike Preuss, and Heike Trautmann

VLR: A Memory-Based Optimization Heuristic 151
Hansang Yun, Myoung Hoon Ha, and Robert Ian McKay

**Classifier Systems, Differential Evolution
 and Swarm Intelligence**

A Differential Evolution Algorithm for the Permutation Flowshop
 Scheduling Problem with Total Flow Time Criterion 161
Valentino Santucci, Marco Baiocchi, and Alfredo Milani

A Taxonomy of Heterogeneity and Dynamics in Particle Swarm
 Optimisation 171
Harry Goldingay and Peter R. Lewis

Derivation of a Micro-Macro Link for Collective Decision-Making
 Systems: Uncover Network Features Based on Drift Measurements 181
Heiko Hamann, Gabriele Valentini, Yara Khaluf, and Marco Dorigo

Messy Coding in the XCS Classifier System for Sequence Labeling 191
Masaya Nakata, Tim Kovacs, and Keiki Takadama

Reevaluating Exponential Crossover in Differential Evolution 201
Ryoji Tanabe and Alex Fukunaga

An Extended Michigan-Style Learning Classifier System for Flexible
 Supervised Learning, Classification, and Data Mining 211
Ryan J. Urbanowicz, Gediminas Bertasius, and Jason H. Moore

Coevolution and Artificial Immune Systems

A Cooperative Evolutionary Approach to Learn Communities
 in Multilayer Networks 222
Alessia Amelio and Clara Pizzuti

Novelty Search in Competitive Coevolution	233
<i>Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen</i>	
An Immune-Inspired Algorithm for the Set Cover Problem	243
<i>Ayush Joshi, Jonathan E. Rowe, and Christine Zarges</i>	

Constraint Handling

Natural Gradient Approach for Linearly Constrained Continuous Optimization	252
<i>Youhei Akimoto and Shinichi Shirakawa</i>	
Evolutionary Constrained Optimization for a Jupiter Capture	262
<i>J�er�mie Labroqu�ere, Aur�elie H�eritier, Annalisa Riccardi, and Dario Izzo</i>	
Viability Principles for Constrained Optimization Using a (1+1)-CMA-ES	272
<i>Andrea Maesani and Dario Floreano</i>	

Dynamic and Uncertain Environments

On the Life-Long Learning Capabilities of a NELLI*: A Hyper-Heuristic Optimisation System	282
<i>Emma Hart and Kevin Sim</i>	
Adaptation in Nonlinear Learning Models for Nonstationary Tasks	292
<i>Wolfgang Konen and Patrick Koch</i>	
On the Effectiveness of Sampling for Evolutionary Optimization in Noisy Environments	302
<i>Chao Qian, Yang Yu, Yaochu Jin, and Zhi-Hua Zhou</i>	

Estimation of Distribution Algorithms and Metamodelling

Evolving Mixtures of n -gram Models for Sequencing and Schedule Optimization	312
<i>Chung-Yao Chuang and Stephen F. Smith</i>	
A Study on Multimemetic Estimation of Distribution Algorithms	322
<i>Rafael Nogueras and Carlos Cotta</i>	
Factoradic Representation for Permutation Optimisation	332
<i>Olivier Regnier-Coudert and John McCall</i>	
Combining Model-Based EAs for Mixed-Integer Problems	342
<i>Krzysztof L. Sadowski, Dirk Thierens, and Peter A.N. Bosman</i>	

A New EDA by a Gradient-Driven Density	352
<i>Ignacio Segovia Domínguez, Arturo Hernández Aguirre, and S. Ivvan Valdez</i>	
From Expected Improvement to Investment Portfolio Improvement: Spreading the Risk in Kriging-Based Optimization	362
<i>Rasmus K. Ursem</i>	
Distance Measures for Permutations in Combinatorial Efficient Global Optimization	373
<i>Martin Zaefferer, Jörg Stork, and Thomas Bartz-Beielstein</i>	
Genetic Programming	
Boosting Search for Recursive Functions Using Partial Call-Trees	384
<i>Brad Alexander and Brad Zacher</i>	
Compressing Regular Expression Sets for Deep Packet Inspection	394
<i>Alberto Bartoli, Simone Cumar, Andrea De Lorenzo, and Eric Medvet</i>	
Inferring and Exploiting Problem Structure with Schema Grammar	404
<i>Chris R. Cox and Richard A. Watson</i>	
Bent Function Synthesis by Means of Cartesian Genetic Programming	414
<i>Radek Hrbacek and Vaclav Dvorak</i>	
Population Exploration on Genotype Networks in Genetic Programming	424
<i>Ting Hu, Wolfgang Banzhaf, and Jason H. Moore</i>	
Improving Genetic Programming with Behavioral Consistency Measure	434
<i>Krzysztof Krawiec and Armando Solar-Lezama</i>	
On Effective and Inexpensive Local Search Techniques in Genetic Programming Regression	444
<i>Fergal Lane, R. Muhammad Atif Azad, and Conor Ryan</i>	
Combining Semantically-Effective and Geometric Crossover Operators for Genetic Programming	454
<i>Tomasz P. Pawlak</i>	
On the Locality of Standard Search Operators in Grammatical Evolution	465
<i>Ann Thorhauer and Franz Rothlauf</i>	

Recurrent Cartesian Genetic Programming	476
<i>Andrew James Turner and Julian Francis Miller</i>	

Multi-objective Optimisation

An Analysis on Selection for High-Resolution Approximations in Many-Objective Optimization	487
<i>Hernán Aguirre, Arnaud Liefooghe, Sébastien Verel, and Kiyoshi Tanaka</i>	

A Multiobjective Evolutionary Optimization Framework for Protein Purification Process Design	498
<i>Richard Allmendinger and Suzanne S. Farid</i>	

Automatic Design of Evolutionary Algorithms for Multi-Objective Combinatorial Optimization	508
<i>Leonardo C.T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle</i>	

Generic Postprocessing via Subset Selection for Hypervolume and Epsilon-Indicator	518
<i>Karl Bringmann, Tobias Friedrich, and Patrick Klitzke</i>	

A Provably Asymptotically Fast Version of the Generalized Jensen Algorithm for Non-dominated Sorting	528
<i>Maxim Buzdalov and Anatoly Shalyto</i>	

Clustering-Based Selection for Evolutionary Many-Objective Optimization	538
<i>Roman Denysiuk, Lino Costa, and Isabel Espírito Santo</i>	

On the Impact of Multiobjective Scalarizing Functions	548
<i>Bilel Derbel, Dimo Brockhoff, Arnaud Liefooghe, and Sébastien Verel</i>	

Multi-objective Quadratic Assignment Problem Instances Generator with a Known Optimum Solution	559
<i>Mădălina M. Drugan</i>	

Optimized Approximation Sets for Low-Dimensional Benchmark Pareto Fronts	569
<i>Tobias Glasmachers</i>	

Start Small, Grow Big? Saving Multi-objective Function Evaluations . . .	579
<i>Tobias Glasmachers, Boris Naujoks, and Günter Rudolph</i>	

Queued Pareto Local Search for Multi-Objective Optimization	589
<i>Maarten Inja, Chiel Kooijman, Maarten de Waard, Diederik M. Roijers, and Shimon Whiteson</i>	

Distance-Based Analysis of Crossover Operators for Many-Objective Knapsack Problems	600
<i>Hisao Ishibuchi, Yuki Tanigaki, Hiroyuki Masuda, and Yusuke Nojima</i>	
Discovery of Implicit Objectives by Compression of Interaction Matrix in Test-Based Problems	611
<i>Pawel Liskowski and Krzysztof Krawiec</i>	
Local Optimal Sets and Bounded Archiving on Multi-objective NK-Landscapes with Correlated Objectives	621
<i>Manuel López-Ibáñez, Arnaud Liefooghe, and Sébastien Verel</i>	
Racing Multi-objective Selection Probabilities	631
<i>Gaetan Marceau-Caron and Marc Schoenauer</i>	
Shake Them All!: Rethinking Selection and Replacement in MOEA/D	641
<i>Gauvain Marquet, Bilel Derbel, Arnaud Liefooghe, and El-Ghazali Talbi</i>	
MH-MOEA: A New Multi-Objective Evolutionary Algorithm Based on the Maximin Fitness Function and the Hypervolume Indicator	652
<i>Adriana Menchaca-Mendez and Carlos A. Coello Coello</i>	
Empirical Performance of the Approximation of the Least Hypervolume Contributor	662
<i>Krzysztof Nowak, Marcus Mörtens, and Dario Izzo</i>	
A Portfolio Optimization Approach to Selection in Multiobjective Evolutionary Algorithms	672
<i>Iryna Yevseyeva, Andreia P. Guerreiro, Michael T.M. Emmerich, and Carlos M. Fonseca</i>	
Using a Family of Curves to Approximate the Pareto Front of a Multi-Objective Optimization Problem	682
<i>Saúl Zapotecas Martínez, Víctor A. Sosa Hernández, Hernán Aguirre, Kiyoshi Tanaka, and Carlos A. Coello Coello</i>	
Parallel Algorithms and Hardware Implementations	
Travelling Salesman Problem Solved ‘ <i>in materio</i> ’ by Evolved Carbon Nanotube Device	692
<i>Kester Dean Clegg, Julian Francis Miller, Kieran Massey, and Mike Petty</i>	

Randomized Parameter Settings for Heterogeneous Workers in a Pool-Based Evolutionary Algorithm	702
<i>Mario García-Valdez, Leonardo Trujillo, Juan Julián Merelo-Guérvos, and Francisco Fernández-de-Vega</i>	
PaDe: A Parallel Algorithm Based on the MOEA/D Framework and the Island Model	711
<i>Andrea Mambrini and Dario Izzo</i>	
Evolution-In-Materio: Solving Machine Learning Classification Problems Using Materials	721
<i>Maktuba Mohid, Julian Francis Miller, Simon L. Harding, Gunnar Tufte, Odd Rune Lykkebø, Mark K. Massey, and Michael C. Petty</i>	
An Analysis of Migration Strategies in Island-Based Multimemetic Algorithms	731
<i>Rafael Nogueras and Carlos Cotta</i>	
Real-World Applications	
Tuning Evolutionary Multiobjective Optimization for Closed-Loop Estimation of Chromatographic Operating Conditions	741
<i>Richard Allmendinger, Spyridon Gerontas, Nigel J. Titchener-Hooker, and Suzanne S. Farid</i>	
A Geometrical Approach to the Incompatible Substructure Problem in Parallel Self-Assembly	751
<i>Navneet Bhalla, Dhananjay Ipparathi, Eric Klemp, and Marco Dorigo</i>	
Application of Evolutionary Methods to Semiconductor Double-Chirped Mirrors Design	761
<i>Rafał Biedrzycki, Jarosław Arabas, Agata Jasik, Michał Szymański, Paweł Wnuk, Piotr Wasylczyk, and Anna Wójcik-Jedlińska</i>	
Evolving Neural Network Weights for Time-Series Prediction of General Aviation Flight Data	771
<i>Travis Desell, Sophie Clachar, James Higgins, and Brandon Wild</i>	
Random Partial Neighborhood Search for University Course Timetabling Problem	782
<i>Yuichi Nagata and Isao Ono</i>	
Balancing Bicycle Sharing Systems: An Analysis of Path Relinking and Recombination within a GRASP Hybrid	792
<i>Petrina Papazek, Christian Kloimüller, Bin Hu, and Günther R. Raidl</i>	

Multiobjective Selection of Input Sensors for SVR Applied to Road Traffic Prediction	802
<i>Jiri Petrlik, Otto Fucik, and Lukas Sekanina</i>	
Evolving DPA-Resistant Boolean Functions	812
<i>Stjepan Picek, Lejla Batina, and Domagoj Jakobovic</i>	
Combining Evolutionary Computation and Algebraic Constructions to Find Cryptography-Relevant Boolean Functions	822
<i>Stjepan Picek, Elena Marchiori, Lejla Batina, and Domagoj Jakobovic</i>	
A Memetic Algorithm for Multi Layer Hierarchical Ring Network Design	832
<i>Christian Schauer and Günther R. Raidl</i>	
Scheduling the English Football League with a Multi-objective Evolutionary Algorithm	842
<i>Lyndon While and Graham Kendall</i>	
Coupling Evolution and Information Theory for Autonomous Robotic Exploration	852
<i>Guohua Zhang and Michèle Sebag</i>	

Theory

Local Optima and Weight Distribution in the Number Partitioning Problem	862
<i>Khulood Alyahya and Jonathan E. Rowe</i>	
Quasi-Stability of Real Coded Finite Populations	872
<i>Jarosław Arabas and Rafał Biedrzycki</i>	
On the Use of Evolution Strategies for Optimization on Spherical Manifolds	882
<i>Dirk V. Arnold</i>	
Unbiased Black-Box Complexity of Parallel Search	892
<i>Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt</i>	
A Generalized Markov-Chain Modelling Approach to $(1, \lambda)$ -ES Linear Optimization	902
<i>Alexandre Chotard and Martin Holeňa</i>	
Level-Based Analysis of Genetic Algorithms and Other Search Processes	912
<i>Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre</i>	

Maximizing Submodular Functions under Matroid Constraints by Multi-objective Evolutionary Algorithms	922
<i>Tobias Friedrich and Frank Neumann</i>	
On the Runtime Analysis of Fitness Sharing Mechanisms	932
<i>Pietro S. Oliveto, Dirk Sudholt, and Christine Zarges</i>	
Runtime Analysis of Evolutionary Algorithms on Randomly Constructed High-Density Satisfiable 3-CNF Formulas	942
<i>Andrew M. Sutton and Frank Neumann</i>	
Author Index	953

Some Computational Aspects of Robot Kinematic Redundancy

Jadran Lenarčič

Jožef Stefan Institute, Jamova cesta 39
1000 Ljubljana, Slovenia
jadran.lenarcic@ijs.si
<http://www.ijs.si>

Abstract. This paper discusses some computational aspects related to the direct and inverse kinematics problems of serial and robot parallel mechanisms, kinematic singularities, workspace determination, manipulability, and kinematic flexibility. With the focus on kinematic redundancy, an example of the humanoid shoulder presents peculiarities in robot and human motion performing different tasks, such as the manipulation of heavy objects or writing. Redundancy enables to the robot to solve different tasks and in an infinite number of ways. The robot can thus simultaneously solve additional secondary tasks of lower priority.

Keywords: Robot kinematics, direct and inverse kinematics problem, parallel robots, workspace, redundant robots, humanoid robots.

1 Introduction

In their early stage, robot mechanisms were simple serial mechanisms possessing six or fewer degrees of freedom. A significant impulse to the development was given in nineties by a dramatic appearance of parallel mechanisms and later by human-like and animal-like mechanisms. Modern robots are intended for use in uncertain, dynamic and unstructured environments and possess complex morphologies of their mechanisms with large number of degrees of freedom. Kinematic analysis and design of such mechanisms is an extensive topic and represents an immense source of computational problems. In many aspects the numerical complexity of these problems poses insurmountable obstacles for the development of practical solutions in study, control, and design of robots as well as their applications.

Computations in robotics cover a rich spectrum of problems at the junction of mechanics, computer science, engineering, and mathematics [2]. In this paper the emphasis is given to various computational aspects in robot kinematics, its analysis, design and optimisation with respect to different tasks. This is related to the problems of direct and inverse kinematics, kinematic singularities, workspace determination, manipulability, and kinematic flexibility. Emphasis is given to the motion of redundant robots performing different human tasks. Redundancy gives to the robot an unlimited source of freedom to solve different tasks and in an infinite number of ways.

2 Basic Problems in Robot Kinematics

The robot mechanism is usually treated as a system of rigid bodies (called links) interconnected by joints [1]. The mechanism can be serial, parallel or serial and parallel combined together, the chain can be open, closed or branched. The layout of the links and joints within a kinematic chain determines the motion properties of the mechanism. A basic property of a mechanism is its ability to move. One of the links is usually fixed to the ground, the moving links of the mechanism are accomplishing various tasks, such as grasping, picking and placing of the objects. From a mathematical point of view, the degrees of freedom within a mechanism equals the minimum number of the mechanism's independent joint variables. These n variables, which are related to the angular or linear displacements in joints, are referred to as joint (generalised or internal) coordinates and are denominated by the vector \mathbf{q} of size n .

The basic equations in robot kinematics describe the position and orientation of the mechanism's links, along with velocities and accelerations of the links. In non-redundant mechanisms, the position and orientation of the terminal link (end effector), determines the pose of each of the previous links of the mechanism. The variables representing the end effector pose are referred to as the task (Cartesian or external) coordinates of the mechanism. They are denominated by the vector \mathbf{p} of size m .

2.1 Direct Kinematics

As is well known, in serial mechanisms the joint and the task coordinates are connected by a system of non-linear equations

$$\mathbf{p} = \mathbf{p}_q(\mathbf{q}), \quad (1)$$

where \mathbf{p}_q is the corresponding vector function in which the joint coordinates typically appear as the arguments of trigonometric functions [2]. These equations can be developed systematically by using homogeneous matrices or other approaches in classical mechanics.

When the joint coordinates appear in the kinematic equations as independent variables, we are dealing with the direct kinematics problem. In serial mechanisms, the direct kinematics does not represent any particular difficulty. For a given vector of joint coordinates \mathbf{q} we can always obtain a unique real solution for the vector of task coordinates \mathbf{p} without regard to their sizes n and m .

The differential linearised, the so-called Jacobian, form of Eq. (1) is given by the following system of equations:

$$d\mathbf{p} = \mathbf{J}_p(\mathbf{q})d\mathbf{q}, \quad (2)$$

where $\mathbf{J}_p(\mathbf{q})$ is the well known Jacobian matrix of the size $m \times n$, and $d\mathbf{q}$ and $d\mathbf{p}$ are increments (or velocities) of joint and task coordinates respectively.

2.2 Inverse Kinematics

When the task coordinates appear in the kinematic equations as independent variables, we are dealing with the inverse kinematics problem. However, an explicit expression of the form

$$\mathbf{q} = \mathbf{q}_p(\mathbf{p}) \quad (3)$$

can only be obtained for especially simple mechanisms, for example Cartesian mechanisms.

In solving the inverse kinematics problem of serial mechanisms we have to face a series of difficulties associated with the non-existence of real solutions, multiple solutions, kinematic singularities, non-existence of closed-form analytical solutions and periodic solutions (in this section we consider only mechanisms where $n = m$, other cases are discussed later):

Non-existence of real solutions – In general, the inverse kinematics problem has a solution inside an interval of values of the task coordinates related to the so-called reachable workspace. If the given task coordinates are out of the reach of the mechanism, no real solution for the joint coordinates exists;

Multiple solutions – In general, for values of task coordinates within the reach of the mechanism there exist multiple solutions for the joint coordinates. They typically appear in pairs. Combinations of joint coordinates representing solutions to the inverse kinematics problem are referred to as configurations. The largest possible number of solutions in serial mechanisms is 2^{n-1} ;

Kinematic singularities – For some values of task coordinates the number of possible configurations is reduced. This can happen for a single point in task coordinates or for a region. This is a kinematic singularity and is a consequence of several solutions merging into a single solution. The kinematic singularity of a robot mechanism is associated with the singularity of the Jacobian matrix (2). It is a mathematical phenomenon resulting in a decreased mobility of a mechanism;

Non-existence of closed-form solutions – Some systems of kinematic equations (1) do not have closed-form solutions to the inverse kinematics problem and the exact solution cannot be obtained. In such cases the solution can only be found with numeric iterative approaches, which may not converge, and as well, may not find all possible solutions;

Periodic solutions – Since equations (1) are trigonometric, there are an infinite number of equivalent periodic solutions for joint coordinates that are rotational. In robot control the joint coordinates expressed as functions of time must be continuous and cannot be allowed to skip from one period into another.

The inverse kinematics of a serial mechanism cannot be solved algebraically for the joint coordinates, explicitly expressed as functions of external coordinates, when the mechanism has more than three degrees of freedom, except in special cases. However, even when an algebraic solution to the inverse kinematics problem exists, searching for the algebraic solution can be rather difficult. The usual approach is based on a transformation of trigonometric equations (1) into a system of polynomial equations, which transforms the problem into searching for the roots of polynomial equations.

A numerical solution to the inverse kinematics problem is used when the system of equations (1) has no algebraic solution. Numerical approaches are well suited for computer programming. Numerous methods exist, among them the most well-known is the Newton-Raphson method and its numerous variations. It is based on the inverse expression of the system (2) as follows:

$$d\mathbf{q} = \mathbf{J}_p^{-1}(\mathbf{q})d\mathbf{p}, \quad (4)$$

which is valid only when the Jacobian matrix is not singular. Moreover, difficulties with numerical methods are associated with the number of iterations, convergence, and with the fact that they cannot provide all possible solutions.

3 Kinematic Redundancy

A mechanism that has too many degrees of freedom with respect to the number of the task coordinates, when

$$n > m, \quad (5)$$

is referred to as kinematically redundant and the difference $D = n - m$ is defined as the degree of redundancy. Kinematic redundancy does not represent any difficulty in solving the direct kinematics problem. The situation becomes more complicated when solving the inverse kinematics problem when the number of unknowns is greater than the number of equations. The kinematic equations (1) and (2) are under-constrained and there exist an infinite number of solutions which belong to the given values of task coordinates. This enables to the robot to solve a given task in infinite number of ways. The kinematic redundancy is normal and frequent phenomenon in human and animal motion.

The kinematic redundancy represents a mathematical complication in solving the inverse kinematics problem, however, a redundant robot, which possesses too many degrees of freedom for the execution of the primary task, can execute additional secondary tasks, such as performing a movement with minimum energy consumption, avoiding obstacles, optimising dexterity or exploiting mechanical advantage. We can calculate the secondary task coordinates \mathbf{s} as follows:

$$\mathbf{s} = \mathbf{s}_q(\mathbf{q}), \quad (6)$$

$$d\mathbf{s} = \mathbf{J}_s(\mathbf{q})d\mathbf{q}, \quad (7)$$

where the number of the secondary task coordinates is l and is arbitrary.

3.1 Task Priority Approach

Resolving the kinematic redundancy is challenging in mathematical and practical terms. The most common in robotics is the so-called task priority approach [5]. This is based on the minimisation of joint displacements relative to the primary and to the secondary task coordinates, where the primary task coordinates possess a higher priority than the secondary task coordinates. Since $n > m$, the primary

Jacobian matrix is rectangular and an independent solution for the primary task uses its generalised inverse

$$\mathbf{J}_p^+ = \mathbf{A}^{-1} \mathbf{J}_p^T (\mathbf{J}_p \mathbf{A}^{-1} \mathbf{J}_p^T)^{-1} \quad (8)$$

and then

$$d\mathbf{q} = \mathbf{J}_p^+(\mathbf{q})d\mathbf{p}. \quad (9)$$

Here \mathbf{A} is a non-singular positive definite matrix of weights.

Similarly, since $l \neq n$, we can form the independent solution of the secondary task as follows:

$$\mathbf{J}_s^+ = \mathbf{J}_s^T (\mathbf{J}_s \mathbf{J}_s^T)^{-1} \quad (10)$$

and then

$$d\mathbf{q} = \mathbf{J}_s^+(\mathbf{q})d\mathbf{s}. \quad (11)$$

Here, the matrix of weights is a unity matrix.

The priority of tasks is assured by the orthogonal complement (null space projector) of the primary Jacobian matrix \mathbf{J}_p ,

$$\mathbf{N}_p = \mathbf{I} - \mathbf{J}_p \mathbf{J}_p^T. \quad (12)$$

Hence, the combined task-priority solution where the secondary task is subordinated to the primary task is as follows:

$$d\mathbf{q} = \mathbf{J}_p^+(\mathbf{q})d\mathbf{p} + \mathbf{N}_p \mathbf{J}_s^+(\mathbf{q})d\mathbf{s}. \quad (13)$$

Here, the solution of the secondary task does not disturb the execution of the primary task. Unfortunately, the execution of the primary task does interfere with the execution of the secondary task and, therefore, numerous improvements of equation (13) have been proposed, such as the optimisation of the matrix of weights \mathbf{A} .

3.2 Measure for Kinematic Redundancy

The self-motion of a redundant mechanism is defined as the displacement of the mechanism which corresponds to unchanging values of the primary task coordinates. It is a tool which enables the redundant mechanism to execute the secondary task along with the primary task. The range of the self-motion can be represented as a subspace in joint coordinates, also known as the kinematic flexibility. It can be quantified as a D -parametric subspace in the n -dimensional space of joint coordinates and can, therefore, serve as a measure of kinematic redundancy [2]. It varies as a function of task coordinates depending on the structure of the mechanism. In spite of its obvious importance in the efficient exploitation of kinematic redundancy, it is still poorly understood and explored. It is because its determination is computationally difficult.

An example is a planar 3R mechanism whose primary task is to position the end-effector. In this case $n = 3$, $m = 2$ and $D = 1$. Thus, the self-motion is

1-parametric and can be visualised by a curve in the 3-dimensional space of the joint coordinates. It can be observed that in some regions of the task coordinates the mechanism possesses a single closed-loop self-motion curve, while in other regions two open-loop curves exist. To reconfigure the mechanism from one self-motion curve to another it is necessary to violate the primary task. Because of this property it can occur that the desired solution of the secondary task is possible, however only along a region of the curve which is not reachable by the mechanism.

4 Kinematic Singularity and Manipulability

Kinematic singularity is mathematically associated with the singularity of the Jacobian matrix in equation (2). At a kinematic singularity, a serial mechanism behaves as if at least one degree of freedom has been lost. When a mechanism approaches a kinematic singularity with a finite velocity in task coordinates, components of the velocity vector in joint coordinates grow to infinite values.

The regions of kinematic singularity, which can be isolated points, curves or surfaces, can be expressed either in the space of task coordinates or joint coordinates. For a long time the opinion prevailed that the kinematic singularity represents the border between two or several configurations of a mechanism. More recent investigations have shown that such mechanisms represent exceptions. A mechanism, where the neighbouring axes are inclined one with respect to other, can move from one configuration into another without going through the singularity.

Kinematic singularities thus represent a significant limitation in robot motion and introduce numerous computational problems in their vicinity especially when gradient-type numerical methods are used to solve the inverse kinematics problem. A lot of research has been devoted to the study of kinematics singularities and how they effect the robot motion in particular in terms of how they could be avoided in the process of robot design and control.

More recent investigations have shown also the positive side of kinematic singularities. In a singular configuration, in the direction of the limited motion, the external force is transformed into zero joint torques. By the use of a proper and more intelligent control, the robot can gain mechanical advantage. In nature, humans and animals use kinematic singularities to minimise the fatigue, for example in lifting weights or walking. In bio-inspired and humanoid robotics, the understanding of kinematic singularities will, therefore, play an important role.

When the displacements in joint coordinates are of equal length and lie on the surface of an n -dimensional sphere

$$d\mathbf{q}^T d\mathbf{q} = \epsilon, \quad (14)$$

the corresponding displacement vectors of external coordinates $d\mathbf{p}$ lie on a surface of an m -dimensional ellipsoid, called the manipulability ellipsoid [7]. The eigenvectors of $\mathbf{J}_p^T \mathbf{J}_p$ are the principal axes of this ellipsoid and the lengths are the corresponding singular values of matrix \mathbf{J}_p . The manipulability is a scalar

and is defined as the product of the singular values of \mathbf{J}_p . Hence, the manipulability brings the information about the volume of the manipulability ellipsoid and therefore measures the manipulation properties of the mechanisms in a given configuration with respect to other configurations.

In matrix algebra the singular values represent the rank of the matrix. If at least one of the singular values is zero, the matrix is singular and the robot end-effector cannot move in at least one direction. The shape of the manipulability ellipsoid is, therefore, equally important. A round ellipsoid suggests that the mechanism can produce equal velocities in all directions, which can be advantageous in applications such as dexterous manipulation. Flat ellipsoids suggest that in some directions the velocity is reduced, which is not necessarily a negative property because in these directions the mechanism produces large forces. This property cannot be quantified with the measure of manipulability. Another measure is of special interest, this is the ratio between the minimal and the maximal singular value. It is called the kinematic index and is a normalised magnitude which describes the roundness of the manipulability ellipsoid. The self-motion in redundant mechanisms enables to change the size and shape of manipulability ellipsoids within a given range in the same position and orientation of the end-effector. Such robots can easily adapt to different task requirements.

Unfortunately, such a general definition and usage of manipulability ellipsoids is not without its problems. If a mechanism is a mixture of different joint coordinates, translations and rotations, as well as different task coordinates, such as positions and orientations, it leads to undesired combination of units, making the comparison between mechanisms difficult and questionable. The manipulability is also size-dependent and can only be useful as a relative measure comparing the same mechanism or very similar mechanisms in different situations.

5 Robot Workspaces

From a broader point of view, the robot workspace is a region which is reachable by a selected point of the mechanism and in which the mechanism possesses certain properties, for instance the ability to reach a prescribed velocity or carry a load. The computation and visualisation of robot workspaces is extremely time consuming and requires an enormous amount of computations. In industrial practice, the so-called reachable and dexterous workspaces are usually used. The volume and form of these are among the most useful criteria to represent and evaluate the reachability of a mechanism [6].

The reachable workspace is the region encompassing all positions which can be reached by a selected end-point of a mechanism, regardless of the orientation of the robot end-effector. This is the basic property of the mechanism, which for short is called the reachability. Reachability appears an indispensable information, however, its significance is rather irrelevant. Namely, if a mechanism is able to reach a certain point in space, this does not mean that a required task can be accomplished at this point.

The dexterous workspace is a region encompassing all positions which can be reached by a selected end-point with all possible orientations of the last

segment or end-effector. In contrast with the reachable workspace the dexterous workspace also reflects the ability of a mechanism to arbitrarily orient the last segment in a selected position. It is a subspace of the reachable workspace.

In the most general case the workspace of a mechanism is determined by computing the position of the selected point on the mechanism and the property which is intended to be examined (such as the load capacity or velocity) using the required kinematic equations for all possible values of joint coordinates. The procedure is executed in such way that the domain of each particular coordinate is divided into a finite number of discrete values. When each coordinate is divided in r intervals, the number of computations is r^n . For a standard mechanism this results in thousands of billions of computations, therefore it is important to properly formulate the involved equations and to minimise the number of arithmetical operations. Different postulates of computational geometry can effectively be applied depending on the mechanism's structure.

6 Parallel Mechanisms

Parallel robot mechanisms are characterised by kinematic structures where the rigid body segments are connected by several parallel kinematics chains [4]. One of the segments represents the fixed base and the other segment is mobile and is called the platform. The kinematics chains connecting the platform with the base are called legs. Unlike a serial mechanism, the degrees of freedom of a parallel mechanism is less than the overall number of degrees of freedom contributed by the robot joints. They can even form rigid kinematic structures.

Parallel mechanisms have only recently emerged in industrial robotics but their history is extensive. The most popular example is the Stewart platform designed in 1965 as a flight simulator. The mechanism, where the mobile platform is controlled by six actuated legs, is still referred to as Stewart platform. In practice, robots with parallel mechanisms exhibit important static and dynamic advantages as follows:

Load capacity, rigidity, and accuracy – Load capacity, rigidity, and accuracy in positioning and orienting are typically several times better than in comparable serial mechanisms. This is because the base and the platform are connected with several kinematics chains;

Excellent dynamic properties – A parallel kinematic structure allows for all actuators and transmissions to be placed on the base and thus they are not moving. The platform can achieve high velocities and accelerations. Also the resonant frequency of a parallel mechanism is orders of magnitude higher;

Simple construction – As only the passive part of the mechanism is mobile, the construction of the mechanism is simpler and less expensive.

Nevertheless, the applicability of parallel mechanisms is limited because of the following main reasons:

Small workspace – The workspace of a parallel mechanism is an intersection of the workspaces of particular legs and is thus significantly reduced. Parallel mechanisms cannot avoid obstacles in their workspace;

Complex kinematics – The calculation and analysis of kinematics in parallel mechanisms is extremely complex and lengthy. The equations are strongly coupled in joint and in general also in task coordinates;

Kinematic singularities – Parallel mechanisms in singularities gain degrees of freedom, which cannot be controlled. While singular poses in serial mechanisms may be nuisance, singular poses of parallel mechanisms result in uncontrolled motion and may be catastrophic.

Parallel mechanisms in combination with kinematic redundancy are common in nature. The human arm, for example, is a system of many parallel subsystems interacting between each other. Parallel mechanisms open an immense amount of research problems and most of them represent highly complex computational challenges associated with their control and design. Thus, the complexity of parallel mechanisms becomes evident in their mathematical analysis.

In parallel mechanisms the calculation of the direct kinematics problem is typically much more complicated than the calculation of the inverse kinematics. The Stewart platform possess one solution to the inverse kinematics problem and 40 solutions to the direct kinematics problem. This was discovered after many years of investigations and finally proven in the middle of nineties. To our knowledge there has been no investigation on how the parallel robot can move from one solution to another without violating its mechanical limitations or intersection of its legs.

7 Example of Bio-Inspired Robot Mechanism

In humans and animals, the muscles stretch over the joints and together with bones create various kinematic structures. The shoulder complex is one of the most illustrative examples. In [3], the human shoulder is modelled as a parallel mechanism of four degrees of freedom associated with the shoulder girdle (Fig.1 - left), and as three intersecting perpendicular rotations associated with the glenohumeral joint (Fig.1 - middle and right).

Hence, the humanoid shoulder mechanism possesses 7 degrees of freedom, where the one related to the expansion of the shoulder girdle is dependent on the

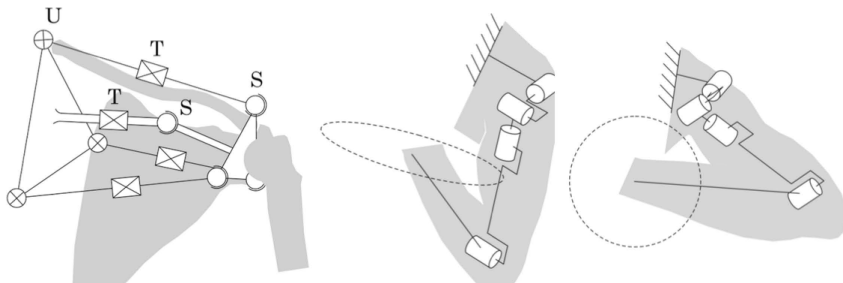


Fig. 1. Humanoid shoulder

girdle inclinations. Typically, the primary task of the human arm is considered the positioning and orienting of the hand in space. The self-motion of the arm is then expressed as the rotation of the centre of the elbow around an axis connecting the centre of the glenohumeral joint and the centre of the wrist. Despite of the limited amount of this rotation, the arm can drastically change the geometry of the manipulability ellipsoid. When the arm is aligned with the trunk and the manipulability ellipsoid is flat (Fig.1 - middle), the arm most efficiently resists to vertical forces. When the elbow is lifted to the height of the shoulder (Fig.1 - right), the manipulability ellipsoid becomes completely round enabling equal displacements in all directions, which can be useful in writing.

8 Conclusions

Various computational aspects in the robot kinematics are given. These are associated to direct and inverse kinematics problems of serial and parallel mechanisms, kinematic singularities, workspace determination, manipulability, and kinematic flexibility. In the end, an example of the humanoid shoulder performing different tasks, such as the manipulation of heavy objects or writing, is briefly discussed. It can be seen, how the kinematic redundancy enables to the robot to solve a primary task in different ways associated with different secondary tasks of lower priority.

References

1. Angeles, J.: *Fundamentals of Robotic Mechanical Systems*. Springer, New York (2007)
2. Lenarčič, J., Bajd, T., Stanišić, M.M.: *Robot Mechanisms*. Springer, Dordrecht (2013)
3. Lenarčič, J., Stanišić, M.M.: Humanoid Shoulder Complex and the Humeral Pointing Kinematics. *IEEE Trans. Robot. Autom.* 19, 499–507 (2003)
4. Merlet, J.-P.: *Parallel Robots*. Springer, Dordrecht (2006)
5. Nakamura, Y., Hanafusa, H., Yoshikawa, T.: Task-Priority Based Redundancy Control of Robot Manipulators. *Int. J. Robot. Res.* 6(2), 3–15 (1987)
6. Sciavicco, L., Siciliano, B.: *Modeling and Control of Robot Manipulators*. Springer, London (2000)
7. Yoshikawa, T.: Manipulability of Robotic Mechanisms. *Int. J. Robot. Res.* 4(2), 3–9 (1985)

Power Distribution Network Reconfiguration by Evolutionary Integer Programming

Kaifeng Yang¹, Michael T.M. Emmerich¹, Rui Li¹,
Ji Wang², and Thomas Bäck¹

¹ LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

² Central South University, 410083 Changsha, Hunan Province, China

Abstract. This paper presents and analyses new metaheuristics for solving the multiobjective (power) distribution network reconfiguration problem (DNRP). The purpose of DNRP is to minimize active power loss for single objective optimization, minimize active power loss and minimize voltage deviation for multi-objective optimization.

A non-redundant integer programming representation for the problem will be used to reduce the search space size as compared to a binary representation by several orders of magnitudes and represent exactly the feasible (cycle free, non-isolated node) networks. Two algorithmic schemes, a Hybrid Particle Swarm Optimization - Clonal Genetic Algorithm (HPCGA) and an Integer Programming Evolution Strategy (IES), will be developed for this representation and tested empirically.

Conventional algorithms for solving multi-objective DNRP are converting the multiple objective functions into a single objective function by adding weights. However, this method cannot capture the trade-offs and might fail in case of a concave Pareto front. Therefore, we extend the HPCGA and IES in order to compute Pareto fronts using selection procedures from NSGA-II and SMS-EMOA. The performance of the methods is assessed on large scale DNRPs.

Keywords: Power Distribution Network Reconfiguration, Integer Programming, Particle Swarm Optimization, Clonal Genetic Algorithm, Evolution Strategies, Multiobjective Optimization.

1 Introduction

With the sustainable development of economy, there is an increasingly high demand for the quality and reliability of the electricity supply in every industry. Power distribution network reconfiguration is an important method of optimizing the distribution system, which is significant to enhancing the security, the efficiency, and the reliability of the system. There are two types of switches in a power network system: normally closed switches and normally open switches. See Figure 1, for an example of a power distribution network configuration, the 119 bus system [1], where the black solid lines represent normally closed switches, and red dashed lines represent normally open switches. Network reconfiguration is the process of changing the topology of the power network by operating these switches for the purpose of minimization of the power loss. Since each switch has

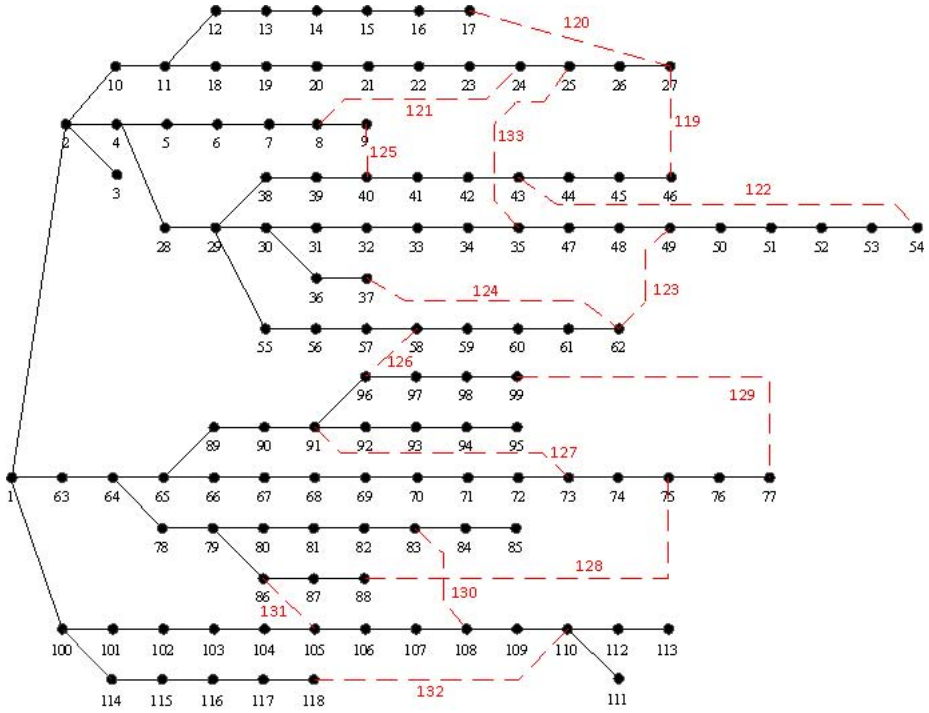


Fig. 1. Initial configuration of the 119-bus test system

two conditions, for a system which has N nodes, there should be 2^{N-1} potential switch configurations. To make sure that all the customers can get electricity and no short circuit exists in the system, there are, however, two constraints for network reconfiguration: no cycles (the radial structure of the network must be maintained in each new structure) and no islands (all the loads must be served).

The problem of finding an optimal distribution network reconfiguration is known to be NP hard, and larger instances with more than 100 nodes cannot yet be solved exactly, such that several metaheuristics were proposed: Zhang et al. [1] developed a tabu search algorithm for real power loss minimal reconfiguration in large-scale distributed systems. Aman et al. [2] proposed to use evolutionary programming (EP) to find the optimal topology of the distributed network for minimizing the real power loss. Rao et al. [3] presented artificial bee colony algorithms for determining the sectionalizing switch to be operated to solve the real power loss minimization problem. Niknam et al. [4] presented an interactive fuzzy satisfying method based on hybrid modified honey bee mating optimization for aggregation-based multiple objective optimization. Kasaei [5] used an ant colony algorithm to solve the optimal network reconfiguration and capacitor placement problem for power loss reduction and voltage profile enhancement in distribution networks.

The contribution of this paper is to discuss concise representations of the problem, discuss fast and reliable evolutionary integer programming solvers,

and extend them for Pareto-based (set-based) multiobjective optimization. The objective functions considered will be power loss and voltage profile enhancement, but as opposed to the study in [5], we are interested in visualizing the trade-off and the robustness by using multiobjective optimization techniques.

2 Problem Description

The network reconfiguration problem in a power distribution system is to find the best configuration of a radial network. The objective functions are the minimization of power loss and the maximization of the network's reliability. Besides, the network has to satisfy certain operating conditions [3].

2.1 Objective Function

The objective function for the **minimization of power loss** can be described as [6]:

$$\min f_{loss} = \sum_{i=1}^b k_i R_i \frac{P_i^2 + Q_i^2}{V_i^2} = \sum_{i=1}^b k_i R_i |I_i|^2 \quad (1)$$

subject to:

$$V_i^{min} \leq V_i \leq V_i^{max} \quad (2)$$

$$I_i \leq I_i^{max}, i = 1, \dots, b \quad (3)$$

Here b is the number of branches and for each branch $i \in \{1, \dots, b\}$, R_i is the branch resistance, P_i and Q_i are the active power and the inactive power of a branch terminal i , V_i is the terminal node voltage of branch i , V_i^{min} and V_i^{max} are the minimum and maximum bus voltage of branch i , respectively, k_i is the status variable of i -th switch. If k_i is 0, then switch i is open and if k_i is 1, then switch i is closed. I_i is the branch current and I_i^{max} is the maximum current in branch i .

The objective function for **minimization of voltage deviation** can be expressed as follows [7][8]:

$$\min f_{VDI} = \max\{|1 - U_{min}|, |1 - U_{max}|\} \quad (4)$$

where U_{min} and U_{max} are respectively the minimum and maximum values of bus voltage divided by rated voltage to normalize them to value in $[0, 1]$.

2.2 Feasible Constraint on Network Topology

Recall that, in order to ensure the supply to all nodes and avoid short circuits, the network must be cycle free and not contain isolated nodes. The previous method for verifying whether an individual is feasible or not is based on the topology checks of the structure [9]. This method can be computed fast, but requires some manual parameter settings beforehand.

A design for an automatic **feasibility check** is proposed next. It rests upon the following idea of *cycle free network construction*: suppose there are N nodes in a system, and there should be at least $(N - 1)$ line segments to connect all these nodes. Then the first line segment can connect 2 arbitrary nodes to the system. Any new line segment, which is not the first line segment, can only connect a new node to the existing system, because otherwise a cycle would be introduced.

The feasibility check algorithm is based on the relationship between the number of the nodes and line segments. Suppose there are N_bus buses and N_line branches in the system. Based on the aforementioned, we can easily conclude that: if $N_line > N_bus$, there must be at least one cycle in the structure. Clearly, if $N_line < N_bus - 1$, there must be at least two separated components in the structure. Therefore, N_line must equal $N_bus - 1$, if the structure is feasible. Given $N_line = N_bus - 1$, feasibility is implied by the non-existence of isolated nodes, due to the following: if there is a cycle and only $N_bus - 1$ edges can be used, at least one of the nodes cannot be connected and it will be an isolated node. As detecting isolated nodes is simple, we can now state the following fast and correct algorithm for checking whether a network is cycle free and does not contain isolated nodes:

Algorithm 1. Topology Feasibility Check

Step 1 Verify N_line equals $N_bus - 1$; if not equal, the individual is not feasible, otherwise go to step 2;

Step 2 Verify whether there is a separated component or not. If there exists a separated component, the individual is not feasible, otherwise it is feasible.

3 Power Loss Minimization Algorithms

Two algorithms for minimizing power loss in DNRP will be discussed next, and extended in section 6 to bi-objective optimization methods. All algorithms are based on a concise sequence encoding of the feasible search space.

3.1 Encoding Strategy

The encoding strategy is a main part which influences the efficiency of the algorithm in distribution network reconfiguration. For a genetic algorithm, it is common to use **binary encoding**, and it is also straightforward in DNRP as each switch can be associated with a binary variable. However, considering the topology constraint, many solutions of the search space induced by binary encoding would be infeasible. Instead, we therefore propose to use **sequence encoding** [10], which has the following advantages: 1. It is easy to realize; 2. The probability of generating a feasible solution is high.

The *sequence encoding system* regards each loop (potential cycle) as a gene. In each loop exactly one switch has to be open, as otherwise either there would

be isolated components (two open switches) or there would be short circuits (no open switch). Each loop of the power network is encoded by a natural number, and all the switches are coded sequentially by natural numbers starting from 1. For each gene, the value of this gene is the position of the switch that is open in the loop represented by the gene. Sequence coding strategy can eliminate most of the infeasible individuals from the search space.

Example 1. Figure 2, is an example of the IEEE-16 distribution system [6][11]. Here a connection of two bus nodes is also a loop. There are three loops in this system. Loop 1 is composed by branches 5, 6, 8 and 9; loop 2 is composed by branches 4, 7, 13, 14 and 15; loop 3 is composed by branches 10, 11 and 12. To make sure the structure satisfying the constraints, one branch in each loop should be opened. Therefore, the search space size is $4 \times 5 \times 3 = 60$ using the sequence encoding system. Compared to a search space size of $2^{12} = 4096$ using binary encoding system the search space can be dramatically reduced.

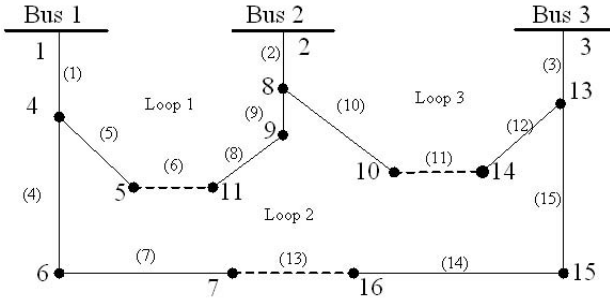


Fig. 2. IEEE-16 distribution system after coding

However, for a real world problem, there could be hundreds or thousands of transformers in one city, and the number of solutions in the search space is very large, even using this sequence coding. For example, in a IEEE-69 system, the binary encoding generates a search space of size $2^{68} \approx 2.95 \times 10^{20}$ and sequence encoding approximately generates a search space size of 1.78×10^6 . For the 119 system (see Figure 1), the search space size is $2^{119} \approx 6.65 \times 10^{35}$ and 1.44×10^{18} , respectively, using binary coding and sequence encoding strategy.

3.2 Hybrid Particle Swarm/Clonal Genetic Algorithm

The Hybrid Particle Swarm Optimization/Clonal Genetic Algorithm (PSO-CGA) is based on two generational transitions (variation and selection steps) that are applied in an alternating manner. The PSO is fast for local optimization, and the CGA is mainly integrated to the PSO in order to prevent premature convergence and increase diversity, e.g., by a special mutation-shift operator. The flowchart of PSO-CGA can be seen in Figure 3.

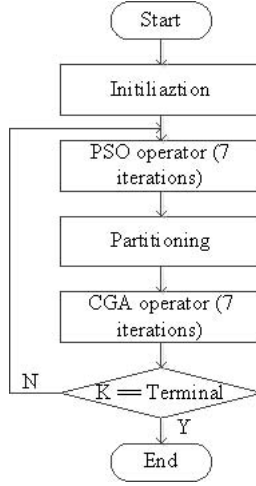


Fig. 3. Flowchart of hybrid PSO-CGA

In PSO it is assumed that the solution space has dimension D , and the population is composed by N particles $X = \{x_1, \dots, x_i, \dots, x_n\}$, the position of the i -th particle is $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})^T$, $P_i^{best} = (P_{i1}^{best}, P_{i2}^{best}, \dots, P_{iD}^{best})^T$ stands for the best known position of particle i , and $g^{best} = (g_1^{best}, g_2^{best}, \dots, g_D^{best})^T$ stands for the best known position of the entire swarm, the velocity of particle i is $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})^T$. Particle x_{iD} updates its velocity and position information by [12]:

$$V_{id}^{k+1} = \omega \times V_{id}^k + c_1 \times r_1 \times (P_{id}^{best} - X_{id}^k) + c_2 \times r_2 \times (g_d^{best} - X_{id}^k) \quad (5)$$

$$X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1} \quad i = 1, 2, \dots, N; d = 1, 2, \dots, D \quad (6)$$

where ω is the inertia factor, c_1 and c_2 are the learning factors, r_1 and r_2 are random real numbers within $[0, 1]$.

Traditionally, clonal genetic algorithms (CGA) [13] use roulette wheel selection. This paper uses another selection mechanism, and this selection mechanism in CGA partitions the parent population P in three subpopulations, say P_B, P_M, P_W , the size of which is fixed by $|P_B| = u_1|P|$, $|P_M| = u_2|P|$, and $|P_W| = u_3|P|$ with $u_1 + u_2 + u_3 = 1$. The new generation is generated in three steps:

Step 1: Select $u_1 \times 100\%$ best offspring for mutation, if there is no improvement after mutation, choose the offspring before the mutation; the general idea is to search for improvements nearby current best solutions.

Step 2: Select $u_3 \times 100\%$ worst offspring for initialization; the general idea behind this mechanism is to maintain the population's diversity.

Step 3: Apply mutation operator for the remaining offspring, the difference between this step 3 and step 1, is that after step 3, the mutated individuals will always be kept, no matter whether there is an improvement or not. This allows slow diffusion away from a local optimum.

3.3 CGA Shift and Mutate Operator

The CGA variation operator is composed by shift and mutation operators [13] [14]. The multi-shift operator adds (or subtracts) the same random offset *Number_Shift* to all genes in a random subset of genes *gene_shift*. A random direction flag *Direction_Flag* is used to decide whether to add or subtract. If the interval boundary gets exceeded the value of the gene is set to the interval boundary.

Example 2. An example for the multi-shift operator is shown in the table below. Where *gene_shift* = [2, 4, 5], and the current individual is [× × 7 11 21].

Parent	<i>Number_Shift</i>	<i>Direction_Flag</i>	Offspring
[× 4 × 11 21]	5	0	[× 9 × 16 26]
[× 4 × 11 21]	5	1	[× 1 × 6 16]

The mutation operator is another main part in CGA. It determines first the genes to be mutated and then sets them to a new random value. A *single-shift* (*single-mutation*) operator shifts (mutates) only a single gene.

4 Integer Programming Evolutionary Strategy

We studied evolution strategies for integer programming (IES) by Rudolph [15], as an alternative search algorithm approach. It features a $(\mu + \lambda)$ -selection scheme, that is λ offspring Q are generated based on μ parents P and the best μ individuals out of $P \cup Q$ form the next parent population. Each individual is created by selecting randomly two parents (sexual case) or more than two parents (panmictic case), applying discrete recombination (choose each gene randomly from one of the parents) or intermediate recombination (averaging) to create a single offspring, which then is mutated. The mutation operator perturbs each gene by the difference of two geometrically distributed pseudo-random numbers. For each gene a step-size ζ_i is maintained and undergoes a mutation, too, which makes it possible for the step-size to adapt. The mutation maps an individual $(\mathbf{X}, \zeta) \in (\mathbb{Z}^n \times (\mathbb{R}^+)^n)$ to its mutant $(\mathbf{X}', \zeta') \in (\mathbb{Z}^n \times (\mathbb{R}^+)^n)$ as follows:

$$\begin{aligned}
 \zeta'_i &= \max\{1, \zeta_i \exp(\tau N_c + \tau' N_i)\}, \quad N_i \sim \text{Normal}(0, 1), \quad N_c \sim \text{Normal}(0, 1) \\
 z_{ij} &= \left\lfloor \frac{\ln(1 - u_{ij})}{\ln(1 - \varphi_i)} \right\rfloor, \quad \varphi_i = 1 - \zeta_i(1 + \sqrt{1 + \zeta_i^2})^{-1}, \quad u_{ij} \sim \text{Uniform}(0, 1), \quad j = 1, 2 \\
 X'_i &= X_i + z_{i1} - z_{i2}, \quad i = 1, \dots, n
 \end{aligned} \tag{7}$$

Since the original geometric distribution is single tailed, Rudolph proposed the use of the difference $z_{i1} - z_{i2}$ and could show that the resulting multivariate distribution has ℓ_1 symmetry, maximal entropy, and infinite support. It features (multiple) self-adaptive mutation step sizes and, for a minimal stepsize greater than zero, global convergence for $t \rightarrow \infty$. In case interval boundaries are exceeded reflection at the interval boundary is used [16]. To prevent stagnation the stepsize is bounded from below by 1. Learning rates τ and τ' determine the speed of stepsize adaptation. See also [16] for a detailed description, default parameters and analysis.

5 Single-objective Optimization Result

The simulation results¹ are based on distribution system 119 [1]. The test system is a 11 kV distribution system with 118 sectionalizing switches and 15 tie switches, and the total power loads are 22709.7 kW and 17041.1 kVAr. The topology of the distribution system 119 is shown in figure 1. All simulations were performed in MATLAB 8.2, CPU: Intel 2 Core 3.16GHz, 2.0 GB DDR RAM (800 MHz). The power flow calculation (to calculate power loss and voltage deviation) is using Newton Method based on MATPOWER [17], and maximum number of iterations is 20, termination tolerance on per unit is 1e-8.

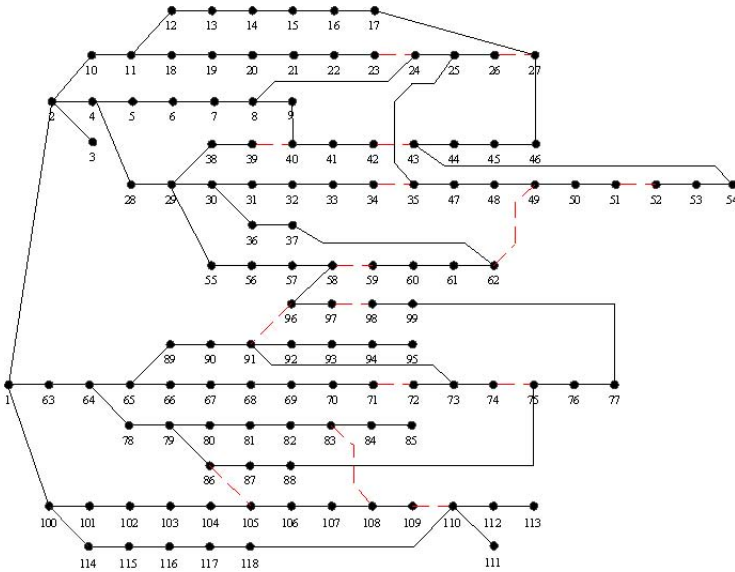


Fig. 4. The best structure

PSO-CGA Hybrid: The PSO-CGA hybrid was run with a population size of 30, and the PSO parameters were $\omega = 0.8$, $c_1 = 2$, $c_2 = 2$, $v_{max} = 4$ (Max speed factor), $v_{min} = -4$ (Min speed factor). For each run, 50 PSO steps (7 generations per step) and 50 CGA steps (7 generations per step) were conducted in alternation. One run takes 15-30 minutes. In the CGA one of the operators, single-shift, single-mutation, multi-shift, or multi-mutation is chosen randomly. In case of multi-shift and multi-mutation the number of genes to be mutated is chosen randomly, too, between 1 and the number of genes. An experiment was designed to find optimal settings for parameters u_1 , u_2 , and u_3 within their bounds $[0, 1]$ and respecting the constraint $u_1 + u_2 + u_3 = 1$. A **design of experiments for mixtures** was applied following parameter setting in [18].

¹ The MATLAB source code of the numerical experiments is available on request by the authors and on <http://natcomp.liacs.nl/index.php?page=code>.

The results (shown in Table 1) of the experimental design are visualized as ternary diagrams (Figure 5). The best results are achieved with a low rate of reinitialization (u_3) and a relatively high rate for u_1 .

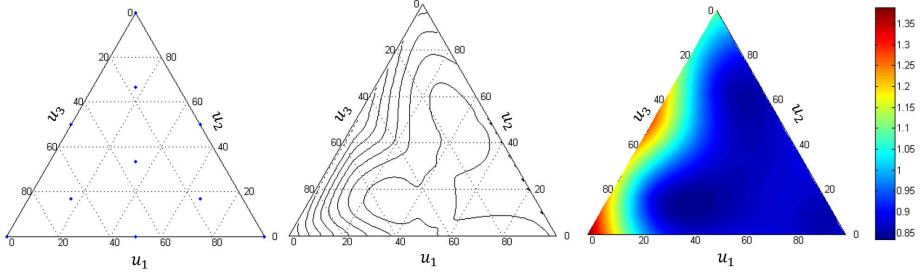


Fig. 5. Mixture design (left) and interpolated results (middle, right) of u_1 (selection), u_2 (diffusion), and u_3 (reinitialization) for CGA-PSO hybrid on Test System 119

Table 1. Experiments on Hybrid PSO-CGA

u_1	1	0	0	0.5	0.5	0	0.333	0.167	0.167	0.666	0.333
u_2	0	1	0	0.5	0	0.5	0.333	0.167	0.666	0.167	0.333
u_3	0	0	1	0	0.5	0.5	0.333	0.666	0.167	0.167	0.333
Exp.1	875.155	1037.9	1329.96	883.828	875.155	1325.33	891.033	891.864	875.155	890.918	874.86
Exp.2	875.155	1085.33	1478.89	875.155	875.155	1120.87	875.155	990.197	875.155	874.86	874.86
Exp.3	875.539	1110.29	1487.06	874.86	890.918	1057.02	875.155	875.155	888.966	875.155	874.86
Exp.4	870.856	1209.17	1138.57	874.86	874.86	1412.98	883.858	887.39	875.155	875.155	875.155
Exp.5	875.155	1068.33	1408.56	875.155	886.437	1366.81	903.49	876.178	876.729	883.641	874.86
Exp.6	875.155	1130.18	1492.56	875.155	875.155	1405.2	875.155	874.86	874.86	874.86	875.155
Exp.7	875.155	1195.5	1492.28	883.858	875.155	1363.57	875.155	869.727	874.86	891.57	875.155
Exp.8	887.505	1081.05	1390.6	877.34	878.96	1186.35	874.86	890.918	874.86	878.209	889.16
Exp.9	874.86	1014.13	1281.5	874.86	874.86	1277.85	878.334	890.918	887.39	874.86	875.155
Exp.10	875.155	1041.54	1384.4	874.86	874.86	1177.81	875.155	875.155	1013.09	869.727	874.86
Min	870.856	1014.13	1138.57	874.86	874.86	1057.02	874.86	869.727	874.86	869.727	874.86
Max	887.505	1209.17	1492.56	883.858	890.918	1412.98	903.49	990.197	1013.09	891.57	889.16
Mean	875.969	1097.34	1388.438	876.993	878.152	1269.38	880.735	892.236	891.622	878.896	876.408
Deviation	4.27671	65.1574	114.1382	3.68630	5.78028	126.08	9.60066	35.3921	43.0215	7.37168	4.482993

Integer Programming Evolution Strategy: The parameters for the IES are as follows: $\mu = 15$, $\lambda = 100$, $\sigma_{initial} = (ub - lb)/8$, $ub_{sigma} = (ub - lb)/3$, $lb_{sigma} = lb$, $\tau = \frac{1}{\sqrt{2n_z}}$ and $\tau' = \frac{1}{\sqrt{2\sqrt{n_z}}}$, where ub and lb is the upper bound and lower bound vector of the variables, $N = 15$ in case of the 119 system. Each run had 100 iterations and took about 5-10 minutes. For the IES preliminar experimentation showed that the recombination type was a crucial choice. Table 2 shows results of different recombination methods based on the Integer ES. We tested four recombination types: none (1), discrete (2), panmictic discrete (3), and intermediate (4); see [19]. The best result that was found was with a discrete recombination on the object variables, and a panmictic discrete recombination on the step size.

Table 2. Experimental results for the 119 system based on ES

		s. _X = 1	s. _X = 2	s. _X = 3	s. _X = 4
s. _ξ = 1	Min	939.6829	893.6304	935.478	930.895
	Max	1302.274	1687.879	1556.437	1904.853
s. _ξ = 2	Min	881.8714	874.8604	869.7271	878.3646
	Max	1083.146	978.1069	974.6982	1048.195
s. _ξ = 3	Min	875.155	894.2617	878.3646	876.0975
	Max	1009.264	1088.795	1120.487	1090.32
s. _ξ = 4	Min	912.3288	1103.351	950.3428	1059.776
	Max	3666.716	7029.378	3071.888	10456

Both Hybrid PSO-CGA and IES can find the same optimal result, see Table 3. The open switches in best result, whose structure is shown in Figure 4, are 42-43, 26-27, 23-24, 51-52, 62-49, 58-59, 39-40, 91-96, 71-72, 74-75, 97-98, 108-83, 105-86, 109-110 and 34-35. Comparing PSO-CGA and IES, it is found that the former was more robust (better average values) while the latter found better results and converged faster (IES 5-10 min, PSO-CGA 15-30 min). All strategies perform significantly better than PSO-CGA with settings $u_1 = 0, u_2 = 0, u_3 = 1$, which is essentially a trial and error strategy (within the sequence representation). The improvement is by a factor of 1298.0861/869.7271, i.e. the power loss found by PSO-CGA or IES metaheuristics is only ca. 67% of the power loss of a solution found by trial and error for the same number of evaluations.

Table 3. Best results for 119 system

Reference	Before reconfiguration			After reconfiguration based on MATPOWER		
	[1]	[3]	Matpower	[1]	[3]	Hybrid PSO-CAG & IES
Unit: kW	1294.3	1298.09	1298.0861	887.5055	869.7271	869.7271

6 Multiobjective Optimization

We extended IES to a multiobjective optimization algorithm by replacing the selection scheme by that of a multiobjective algorithm, namely the $(\mu + \mu)$ selection of NSGA-II [20] and the $(\mu + 1)$ selection of SMS-EMOA [21] (which has earlier been used also in Pareto archivers [22]). As a second objective voltage deviation was minimized (see equation 4).

As an adaptation, we introduced a variant of SMS-EMOA and NSGA-II with a self-adaptive single step size. Whenever more than five mutations per individual were unsuccessful, the step size was multiplied by a constant factor of 1/1.2 (following the 1/5th success rule). Success of a generation was registered if a new non-dominated solution entered the archive of non-dominated solutions among all solutions encountered so far.

The results (attainment curves) for a population size of $\mu = 30$ and 11 runs per algorithm are shown in Figure 6, where f_1 and f_2 represent voltage deviation and

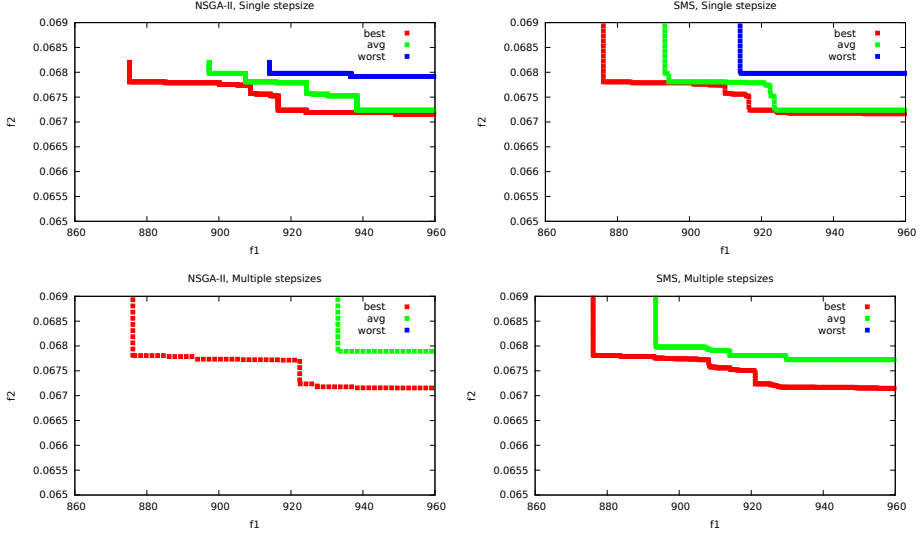


Fig. 6. Best, worst, and average attainment curves for the multiobjective optimization of 119 DNRP

power loss. In the following discussion by Pareto front, we mean the archive of all non-dominated solutions encountered in a single run. SMS-EMOA with a single step size provides the best Pareto fronts in the best case and also in the average case. Interestingly, all strategies find a Pareto front with a concave part. The interpretation of this is that locally there is a strong conflict between power loss minimization and voltage deviation minimization for this problem. However, the range of voltage deviation is relatively small, so that 'from a distance' the Pareto front has a clear knee point region. Solutions in this region can be recommended as good compromise solutions, whereas points located on the flanks of the Pareto front are not recommended as small improvements in one objective will cause large deterioration of the other objective.

7 Conclusions

In this paper two well performing optimization strategies for solving the power distribution network reconfiguration problem have been described and tested on a challenging problem with more than 100 switches. A concise integer representation was chosen and it was demonstrated that it reduces the search space size by many orders of magnitude as opposed to the binary representation used in genetic algorithms so far. In the experiments we focused on finding a good ratio between exploration and exploitation in the PSO-CGA and on choosing a good recombination operator in the IES with self-adaptive mutation. This turned out to be discrete recombination on object and step-size variables. The results clearly show that it is much better to use metaheuristics instead of trial and

error strategies when minimizing power loss. Also multiobjective optimization algorithms, NSGA-II and SMS-EMOA, were applied to compute a Pareto front between voltage deviation and power loss objectives. These objectives turned out to be conflicting in the knee point region but globally, when zooming out, they appear to be complementary. For the future work, it is recommended to test the strategies on a broader set of benchmarks and further investigate multiobjective problem formulations, for instance including objective functions on reliability.

Acknowledgment. Kaifeng Yang acknowledges financial support from China Scholarship Council (CSC), CSC No.201306370037.

References

- [1] Zhang, D., Fu, Z.C., Zhang, L.C.: An improved ts algorithm for loss-minimum reconfiguration in large-scale distribution systems. *Electric Power Systems Research* 77(5), 685–694 (2007)
- [2] Aman, M.M., Jasmon, G.B., Naidu, K., Bakar, A.H.A., Mokhlis, H.: Discrete evolutionary programming to solve network reconfiguration problem. In: *TENCON Spring 2013 Conference*, pp. 505–509. IEEE, Sydney (2013)
- [3] Rao, R.S., Narasimham, S.V.L., Ramalingaraju, M.: Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm. *International Journal of Electrical Power and Energy Systems Engineering* 1(2), 116–122 (2008)
- [4] Niknam, T., Meymand, H.Z., Mojarrad, H.D.: An efficient algorithm for multi-objective optimal operation management of distribution network considering fuel cell power plants. *Energy* 36(1), 119–132 (2011)
- [5] Kasaei, M.J., Gandomkar, M.: Loss reduction in distribution network using simultaneous capacitor placement and reconfiguration with ant colony algorithm. In: *Asia-Pacific Power and Energy Engineering Conference, APPEEC 2010*, pp. 1–4. IEEE, Chengdu (2010)
- [6] Qin, Y.M., Wang, J.: Distribution network reconfiguration based on particle clonal genetic algorithm. *Journal of Computers* 4(9), 813–820 (2009)
- [7] Chiou, J.P., Chang, C.F., Su, C.T.: Variable scaling hybrid differential evolution for solving network reconfiguration of distribution systems. *IEEE Transactions on Power Systems* 20(2), 668–674 (2005)
- [8] Niknam, T.: A new hybrid algorithm for multi-objective distribution feeder reconfiguration. *Cybernetics and Systems: An International Journal* 40(6), 508–527 (2009)
- [9] Wang, S.X., Wang, C.S.: A novel network reconfiguration algorithm implicitly including parallel searching for large-scale unbalanced distribution systems. *Automation of Electric Power Systems* 24(19), 34–38 (2000)
- [10] Ma, X.F., Zhang, L.Z.: Distribution network reconfiguration based on genetic algorithm using decimal encoding. *Transactions of China Electrotechnical Society* 19(10), 65–69 (2005)
- [11] Bi, P.X., Liu, J., Liu, C.X., Zhang, W.Y.: A refined genetic algorithm for power distribution network reconfiguration. *Automation of Electric Power Systems* 26(2), 57–61 (2002)

- [12] Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. *Swarm Intelligence* 1(1), 33–57 (2007)
- [13] Li, M.J., Tong, T.S.: A partheno genetic algorithm and analysis on its global convergence. *Automatization* 25(1), 68–72 (1999)
- [14] Wang, J., Luo, A., Qi, M.J., Li, M.J.: The improved clonal genetic algorithm & its application in reconfiguration of distribution networks. In: *Power Systems Conference and Exposition, PSCE 2004*, pp. 1423–1428. IEEE, New York (2004)
- [15] Rudolph, G.: An evolutionary algorithm for integer programming. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) *PPSN III. LNCS*, vol. 866, pp. 139–148. Springer, Heidelberg (1994)
- [16] Li, R., Emmerich, M., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J., Reiber, J.H.: Mixed integer evolution strategies for parameter optimization. *Evolutionary computation* 21(1), 29–64 (2013)
- [17] Zimmerman, R.D., Murillo-Sánchez, C.E., Thomas, R.J.: Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems* 26(1), 12–19 (2011)
- [18] Anderson, M., Whitcomb, P.: Find the optimal formulation for mixtures (2002), <http://www.statease.com/pubs/chem-2.pdf>
- [19] Bäck, T.: *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, New York (1996)
- [20] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.A.M.T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
- [21] Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181(3), 1653–1669 (2007)
- [22] Knowles, J., Corne, D.: Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation* 7(2), 100–116 (2003)

In Vivo Veritas: Towards the Evolution of Things

Agoston Endre Eiben

VU University Amsterdam, The Netherlands
a.e.eiben@vu.nl

Abstract. The main thesis of the position paper is that in the near future it will be possible to create populations of animate physical objects that undergo evolution in real space and real time. The resulting systems will differ from Evolutionary Computing in two crucial aspects. First, the individuals will be physical rather than digital. This requires reproduction operators for physical objects, which forms an engineering challenge. Second, the evolutionary process will be induced by the autonomous behavior of the individuals themselves, not by some central evolutionary agency that orchestrates selection and reproduction. These differences imply severe challenges for evolutionary algorithm designers because ‘tricks’ that work in *in silico* may not work *in vivo*. However, overcoming these challenges will ignite the development of a new field that combines Evolutionary Computing, Robotics, Artificial Life, and Embodied AI with a great potential for engineering as well as scientific research.

Keywords: Embodied Evolution, Evolutionary Computing, Evolutionary Robotics, Self-reproducing Robots.

1 Introduction

This is a position paper corresponding to the keynote I gave on the 13th International Conference on Parallel Problem Solving from Nature, a.k.a. PPSN 2014, about what I call the Evolution of Things or strongly Embodied Evolution.¹ It builds on the ideas presented in my TEDx talk (<http://tinyurl.com/EibenTEDx>) and the 2012 paper in the Evolutionary Intelligence journal [17]. To avoid a big overlap with these, in this paper I focus on the technical aspects from an evolutionary perspective and illuminate certain challenges and possible solutions based on earlier work of my collaborators and myself.

Perhaps the best way to introduce the underlying vision is to contrast two types of evolutionary processes we know today, programmable artificial evolution in computer models (*in silico*) and real-world natural evolution out in the wild (*in vivo*), cf. Figure 1. Note that “programmable” is meant there in a loose sense. It does not imply the ability to deterministically drive the system to some state. Rather, it means the ability to specify the details of the individuals’ makeup and to prescribe rules for their behavior. The subject of this paper is the new, exciting area of research in the intersection that

¹ The term “embodied evolution” is used in [42] to describe the (distributed) evolution of controllers in a population of physical robots with fixed morphologies. To avoid confusion one could call that system “weakly embodied evolution” and use “strongly embodied evolution” for systems where the bodies evolve as well.

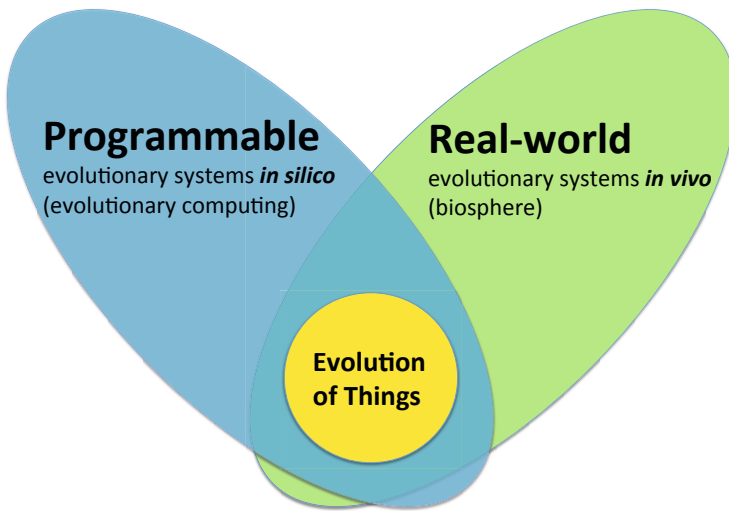


Fig. 1. The Evolution of Things takes place in real space and real time based on (self-)reproducing physical -rather than digital- entities

will feature evolvable artefacts (with designable makeup and behavioral rules) in the physical world. In other words, I will speculate about machines with evolvable bodies and minds.

Natural evolution is the force behind the emergence of Life on Earth, as established in the 19th century [9]. The invention of the computer in the 20th century made it possible to create artificial worlds and actively engineer artificial evolutionary processes in digital spaces. The resulting field, called Evolutionary Computing, was groundbreaking in that it converted evolution from a passive explanatory *theory* meant to explain a past process into an active *tool* meant to create new processes. However, an evolutionary computing process and the underlying computer models only capture the ‘macro-mechanics’ of evolution through a simplified genotype-phenotype mapping and an abstract selection-reproduction loop. The biochemical and physical ‘micro-mechanics’ are ignored and in the current practice the emphasis lies on (ab)using artificial evolution as an optimizer. Nevertheless, by the development of Evolutionary Computing and related areas in Artificial Life we –the research and user communities– have gained much experience about working with artificial evolution. We have learned to construct various forms of evolvable digital objects. We have invented and studied various selection and reproduction mechanisms, including ones that do not exist in Nature, e.g., crossover mechanisms between more than two parents [15]. And we have designed numerous evolutionary algorithms inspired by natural mechanisms, but not limited by constraints of physical or biological reality. All in all, we have developed the know-how to set up and manage artificial evolutionary processes and to use them for solving optimization, design, and modeling problems [2,10,18].

Evolutionary Computing is, well, computing. Producing a new individual in an evolving population is just a matter of creating a new piece of digital code. The same holds

for evolving virtual creatures in Artificial Life [37]. However, as noted in [19], such systems seriously lack “the richness of matter that is a source of challenges and opportunities not yet matched in artificial algorithms”. Going from digital evolutionary systems to physical ones will be a game changer in several ways and will represent a major transition from a historical perspective that brings artificial evolution closer to natural evolution [17].

2 Robots?

The Evolution of Things as I envision takes place in real space and real time, based on (self-)reproducing physical -rather than digital- entities. In order to refine this vision it is useful to distinguish ‘mindless’ things and ‘animate’ things and to state that the idea here is to evolve animate things that can sense, make decisions, and perform actions autonomously. Of course, it is possible to create a system of evolving mindless objects², but the case of animate objects is more interesting and promises more applications. This makes robots, a.k.a. intelligent machines, relevant because robots are physical objects that can sense, make decisions, and perform actions autonomously. One could even extend the traditional notion of robots and postulate that any kind of animate artefact or machine capable of sensing, making decisions, and performing actions autonomously is a robot, regardless of the substrate that determines its physical makeup, i.e., body, and control architecture, i.e., mind.

Carrying this view further we could say that The Evolution of (animate) Things is the same as the evolution of robots, if only we define robots in the broad sense. It could be argued that this definition is too limited because the notion of robots is not broad enough. For instance, chemists working on specially engineered molecules that can self-replicate or biologists trying to strip down living cells to make them programmable may not see their evolving entities as robots, although they could be called ‘things’. Furthermore, it could be incorrect to see molecules as animate entities because do not have the ability to sense and to make decisions. However, this discussion is beyond the scope of this paper, the point I want to make is that willing to evolve animate things naturally leads to Evolutionary Robotics (ER).

Evolutionary Robotics is the combination of evolutionary computing and robotics [6,12,20,33,38,39,41]. ER is a field that “aims to apply evolutionary computation techniques to evolve the overall design or controllers, or both, for real and simulated autonomous robots” [39]. This approach is “useful both for investigating the design space of robotic applications and for testing scientific hypotheses of biological mechanisms and processes” [20]. The field of ER has made much progress over the last decade and a half. A recent overview, cf. [6], summarizes the key insights as follows:

- Manual design of a mobile robot that is autonomous and adaptive is extremely difficult.
- As an alternative, computers can ‘evolve’ populations of robots in a simulator...

² In such a system the objects just passively undergo evolutionary operators executed by some ‘evolution manager’ – quite like the digital individuals in a usual evolutionary algorithm.

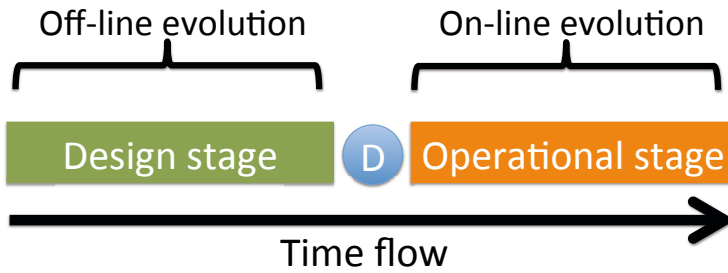


Fig. 2. Workflow of robot design distinguishing the design stage and the operational stage of the robots, separated by the moment of deployment (circle in the middle). Off-line evolution takes place in the design stage and the evolved features (usually the controllers) do not change after deployment. On-line evolution is performed during the operational period, which means that the robot’s features are continually changed by the evolutionary operators.

- This evolutionary approach changes the way we view robotics: ... focus shifts to creating an evolutionary system that continuously designs and manufactures different robots with increasing abilities.

However, as noted in [6] “the use of metaheuristics [i.e., evolution] sets this subfield of robotics apart from the mainstream of robotics research” which “aims to continuously generate better behavior for a given robot, while the long-term goal of Evolutionary Robotics is to create general, robot-generating algorithms”.

From the Evolutionary Computing perspective, ER is a special application area that is different from, say, combinatorial optimization. Somewhat oversimplifying, the main challenge in solving optimization problems with EAs is the ruggedness of the fitness landscape defined by the objective function. For ER applications there are two additional problems: the very weak and noisy link between controllable design details and the target feature(s) and the great variety of conditions / requirements under which a solution should prove good. For example, if we are to evolve NeuralNet controllers for a robot then the NN descriptors (direct or indirect parameters of the NN topology and weights) are the genotypes and the NN controllers form the phenotypes. Unlike in ‘simple’ optimization, these phenotypes cannot be directly evaluated. Rather, it is the robot behavior induced by the given controller that is observed and assessed. Thus, in usual EC we have a 3-step chain: genotype – phenotype – fitness, while in ER the chain is 4-fold: genotype – phenotype – behavior – fitness. Additionally, the behavior depends on many external factors not only on the genotype and the evaluation of a controller requires running the robot for a while under different circumstances. Last but not least, desirable robot behavior is almost never defined by one single skill (except for pure research purposes). For instance, it could be required that the robot performs well in various arenas, under different light conditions regarding its skills for locomotion, collision avoidance, target following, object manipulation, and cooperation with other robots. Consequently, fitness functions in ER are inherently very noisy, very expensive, and multi-objective in terms of behavioral requirements [32].

Since this paper is not meant to be a survey of evolutionary robotics in the following I only consider two features that most ER applications share: 1) the off-line character

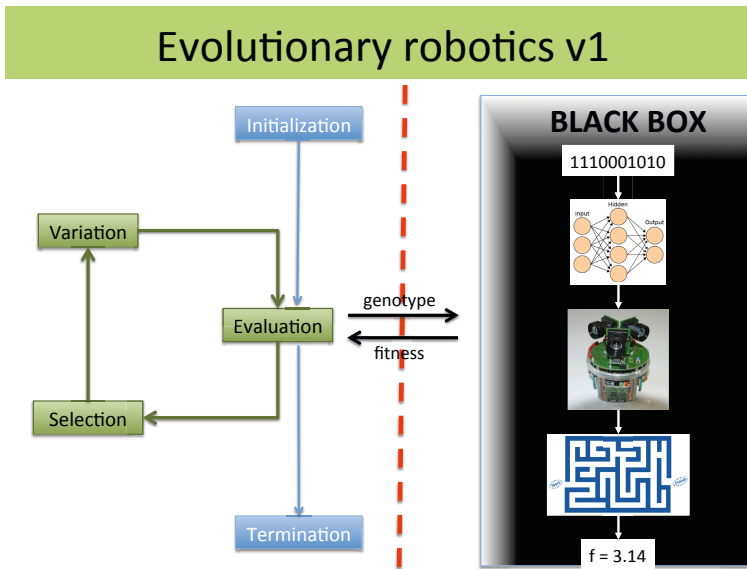


Fig. 3. Most ER applications use a quite straightforward evolutionary algorithm, only the fitness evaluations are special: these are mainly done by a simulator, sometime including occasional evaluations with the hardware in-the-loop

of the evolutionary process and 2) the use of simulators. From this perspective, the Evolution of Things is beyond conventional evolutionary robotics, because it is based on on-line evolution in the real world.

2.1 Evolutionary Robotics Version 1: Off-line Evolution

To illuminate the on-line versus off-line aspect consider Figure 2. The usual approach in evolutionary robotics employs an evolutionary algorithm to find a good controller before the operational period of the robot. The evolutionary algorithm (EA) is quite straightforward, only the fitness evaluations are special: these are done by a simulator, possibly under different starting conditions, cf. Figure 3. When the user is satisfied with the evolved controller, then it is deployed (installed on the physical robot) and the operational stage can start. In general, the evolved controllers do not change after deployment during the operational stage, or at least not by evolutionary operators. Naturally, there are studies that use the ‘hardware in the loop’ for (some of the) fitness evaluations, but this does not change the general workflow illustrated by Figure 3, most importantly, it does not require different kinds of EAs. Let me note that this workflow also applies for most studies that use off-line evolution for evolving morphologies. The huge majority of work in ER falls in this category belonging to the upper half of the table shown in Figure 6.

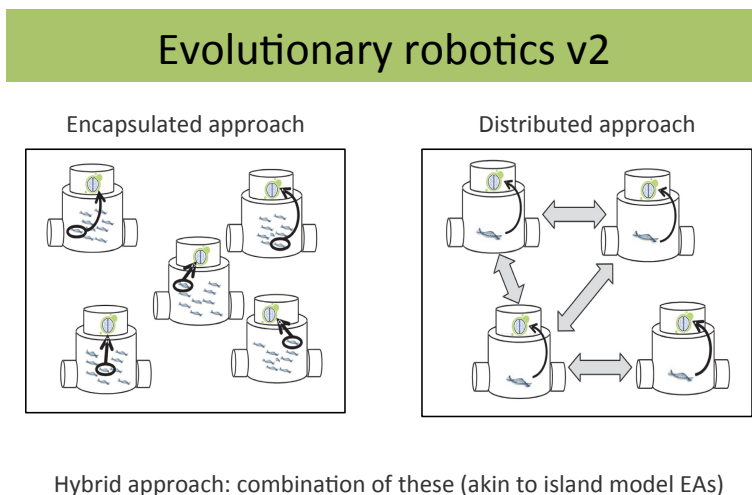


Fig. 4. In on-line ER applications the evolutionary algorithm runs in real-time on the robots themselves. Thin black arrow from the ‘DNA’ to the ‘brain’: the genotype is expressed and the corresponding phenotype (controller) is activated. Fat grey arrow between robots: interaction for mating (recombining genotypes). Fitness evaluations are done on the real hardware and they cannot be repeated for good statistics under the same conditions.

2.2 Evolutionary Robotics Version 2: On-line Evolution of Controllers

In principle, there is an option to apply on-line evolution to evolve robots controllers during the operational period [16]. This implies that evolutionary operators can change the robots’ control software even after deployment. Although this option has already been investigated early on in the history of the field, see for example [35,42], relatively little effort has been devoted to this type of systems. This preference is not surprising, because it is fully in line with the widespread usage of EAs as optimizers, which fits the off-line approach very well. However, natural evolution is not a function optimizer, nor are evolutionary algorithms [11]. The natural role of evolution is that of permanent adaptation and using artificial evolution in this role in a group of robots requires adjustments to the usual EA setup as illustrated in Figure 4. This role is expected to become more and more important in the future of robotics. The advantages of such systems is phrased in [32] as follows:

“Advanced autonomous robots may someday be required to negotiate environments and situations that their designers had not anticipated. The future designers of these robots may not have adequate expertise to provide appropriate control algorithms in the case that an unforeseen situation is encountered in a remote environment in which a robot cannot be accessed. It is not always practical or even possible to define every aspect of an autonomous robot’s environment, or to give a tractable dynamical systems-level description of the task the robot is to perform. The robot must have the ability to learn control without human supervision.”

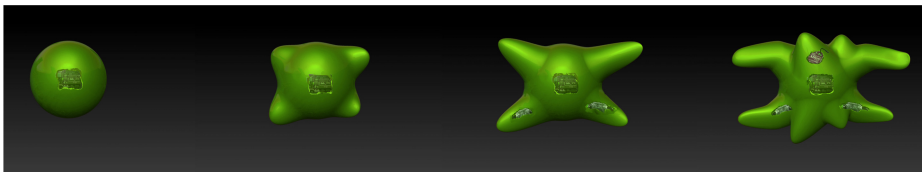


Fig. 5. Artist impression of soft robots with evolving morphologies. Consecutive images illustrate members of different generations from the start (left) to the end (right). Courtesy of Pablo Gil-Cornaro and Claudio Rossi.

From the perspective of an evolutionary algorithm designer such systems are interesting and challenging, because they have a dynamics different from usual EAs and version 1 of ER (ER-v1). For instance, such systems have two kinds of units, the robotic units and the evolutionary units. The robotic units are the physical, pre-engineered, and fixed bodies that contain the computers that run the evolutionary algorithm. However, unlike in ER-v1, these computers can move, interact, and their movements and interactions depend on the evolving controllers. The evolutionary units are the controllers that form the evolving population, they undergo selection and reproduction. These units are digital, flexible, and continually changing. The interactions between these two types of entities has not been studied yet. In terms of the nomenclature introduced in Section 1 this version of ER can be called weakly embodied. One important feature of such systems is that the number of bodies, that is robots, is given and cannot be extended. This implies ‘no-go-areas’ in the search space because a bad guess (a poor controller resulting from an unlucky variation operator) can be ‘lethal’ for the hosting robot body if tested in real hardware. While in usual EC bad guesses are just wasting time, in weakly embodied ER they can waste the robots.

2.3 Evolutionary Robotics Version 3: On-line Evolution of Morphologies (and Corresponding Controllers)

Work concerning the evolution of morphologies is scarce and either not on-line or not physical. That is, existing work is done either in an off-line manner in computer simulations only constructing the evolved robots afterwards, see for instance [30], or in an on-line fashion but in simulation. Papers in this latter category are often positioned within Artificial Life, investigating the evolution of ‘virtual creatures’ [37] or ‘machines’ in general [3], rather than robots in particular. As the question mark in Figure 6 indicates on-line evolution of robot morphologies and the corresponding controllers has not been done yet. The reason is quite obvious: reproduction operators for physical artefacts are much harder to implement than for digital objects. In evolutionary computing there are several mutation and crossover operators for all kinds of genotypes from simple bit-strings to complex decision trees and the construction of the resulting child(ren) is trivial in software. However, doing the same in real hardware is a different story and self-reproducing robots form one of the Grand Challenges for Evolutionary Robotics proposed in [13].





Evolutionary Robotics Categories		
	Controllers	Morphologies
Off-line		
On-line		

Fig. 6. Four categories within Evolutionary Robotics based on what is being evolved (controllers or morphologies) and how it is evolved (off-line or on-line). The sizes of the circles indicate the number of papers in each category. NB Circles are not at real scale.

After this brief overview, the Evolution of Things can be described from an ER perspective: it amounts to strongly embodied evolution, that is, ER of the third type, where morphologies and corresponding controllers evolve on-line in the real world.

3 The Evolution of Things: Why

There are several reasons to be interested in the Evolution of Things [17]. The technology of evolvable robots offers possible applications in the future, where adapting the robot design and/or producing new robots during the operational period without human intervention is important. This can be the case in inaccessible environments, for example, colonies of mining robots that work in extreme depths under the surface of the Earth for extended periods, planetary missions, deep sea explorations, or medical nano-robots acting as ‘personal virus scanners’ inside the human body. Additionally, self-reproducing robots can be evolved with the human in the loop very much like breeding livestock. This can combine the human guidance (user selection) with the creative exploratory power of evolution as used today in *in silico* evolutionary design [4,5]. There are also benefits for scientific investigations including biological research where robots can be used as the substrate to create physical, rather than digital, models of biological systems and to study biological phenomena [21,31,40]. Furthermore, this new technology offers unprecedented opportunities for embodied Artificial Intelligence. In an evolving population of self-reproducing robots minds and bodies can co-evolve in the real world. This eliminates the restriction of working with fixed morphologies and opens the possibility to studying the mind-body problem in a new way [1,7,26,27]. One could say that with the new technology we cannot only study how the body shapes the mind, but also how the mind shapes the body [34].

To illustrate the limitations of studying virtual creatures let me consider an investigation into the “Effects of Evolutionary and Lifetime Learning on Minds and Bodies in an Artificial Society” published in [8]. The study concerns a simulated artificial environment with a population of individuals that have a body as well as a mind. That is, some of their features effect their physical properties, like speed and strength, while other features influence their mental preferences in interacting with the environment and other agents. The paper compares two approaches to adapting these individuals. In the first approach the bodies and the minds develop through evolution, i.e., body features as well as mind features are inheritable, hence evolvable. In the second approach only the bodies evolve and the minds are adapted by lifetime-learning. In both cases the system is purely environment driven without a user-defined quantitative fitness measure. The results indicate that the first approach is able to sustain larger and more stable agent populations and maintain a higher degree of individual success. Furthermore, quite unexpectedly, the two systems differ a lot concerning the kind of bodies that emerge over time. That is, the individuals’ bodies in the last populations reside in completely different segments of the physical feature space under the two regimes even though the environment is the same. This is an interesting outcome, because it means that all other things being equal, the method used for mental development has a strong effect on the development of the physical features.

Unfortunately, it is hard to establish the general relevance of this result which could just be rooted in the properties of the overly simple model of the world, the body features, and the interactions between them. In fact, the system can be physically implausible and violate some laws of physics. The simulated world may differ from the physical one to such an extent that the experimental findings are the opposite of the real world effect. Conducting this or a similar study *in vivo*, in an evolving population of real robots would eliminate these concerns. Phrasing it from a robotics perspective, in a strongly embodied evolutionary system there will be no reality gap anymore.

4 The Evolution of Things: How

To provide the algorithmic underpinning of evolving robots in real-time and real-space a conceptual framework, dubbed the Triangle of Life (ToL), has been proposed recently [14]. The ToL scheme shown in Figure 7 does not make assumptions on the physical substrate of the evolving organisms; these can be (modular) mechatronic robots, soft robots, artefacts with nonconventional bodies and forms of control, even (bio)chemical entities.³ Therefore the ToL does not contain general recipes for the birth / morphogenesis operator shown by the left arrow in Figure 7. How this operator is implemented depends on the given substrate for the robot bodies. With an evolutionary computing analogy the ToL can be perceived as the equivalent of the general evolutionary algorithm loop that captures the main components of one evolutionary cycle without specifying which representation is being used, cf. Fig. 2.2. in [18].

³ The paper [14] illustrated the components of this framework one by one using the modular robots of the Symbrion project. However, Symbrion was not aiming at physically evolving morphologies and the components of the ToL have not been integrated.

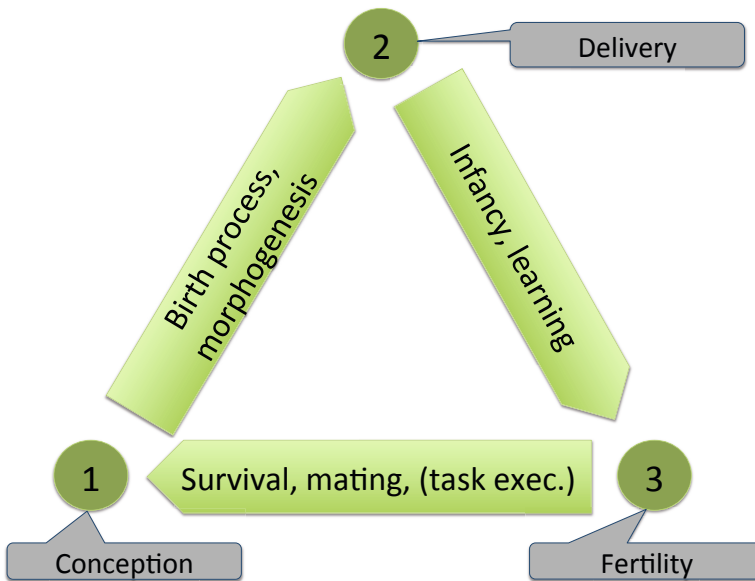


Fig. 7. The Triangle of Life after [14]. The pivotal moments that span the triangle are: 1) Conception: A new genome is activated, construction of a new organism starts. 2) Delivery: Construction of the new organism is completed. 3) Fertility: The organism becomes ready to conceive offspring.

The proverbial Cycle of Life revolves around birth. The ToL framework adopts this stance and defines a life cycle that does not run from birth to death, but from conception (being conceived) to conception (conceiving one or more children). The main idea is generic, the only significant assumption is the genotype-phenotype dichotomy. That is, it is presumed that the evolvable objects as observed ‘in the wild’ are the phenotypes encoded by their genotypes. In other words, the artefacts in question can be seen as the expression of a piece of code called the genome. As part of this assumption it is postulated that reproduction takes place at the genotypic level. This means that the evolutionary operators mutation and crossover are applied to the genotypes (to the code) and not to the phenotypes (to the physical artefacts). Nevertheless, creating new pieces of code by crossover and mutation must be followed by the physical production of the encoded entity by a birth or morphogenesis process. This is the most important distinguishing feature of the type of evolutionary systems that the ToL framework specifies.

Recall that The Triangle of Life framework is agnostic about the birth operator. However, it is important to note that birth should be implemented by a centralized system component, by a ‘Birth Clinic’ that constructs a new organism from a building plan, i.e., from the genome created by recombining/mutating the genomes of the parents. Distributed solutions (‘pregnancy’ or ‘eggs’) must be avoided in favor of a system with a single point of failure that can be used as a kill switch if the evolutionary process needs to be halted. I consider this an important issue of principle and emphasize that all physically embodied evolutionary systems of the future must be designed with a shutdown guarantee.

In my view such systems of evolving robots (in the broad sense) implemented through the ToL framework represent a new class of Artificial Life. Regarding the question how such a system might be used two contrasting applications present themselves. One as an engineering solution to a requirement for multiple robots in extreme unknown or dynamic environments in which the robots cannot be specified beforehand or have to be (re)adjusted to the changing conditions. The other application is scientific. Such artificial life systems could be used to investigate the development of embodied intelligence and new types of evolutionary processes, not so much to model biological evolution, life as it is, but to study life as it could be.

5 Special Algorithmic Challenges

The development of strongly embodied evolutionary systems bears special relevance for the evolutionary computing community. The scientific and technical knowledge regarding artificial evolutionary systems has been accumulated within this community over the last decades. Therefore, evolutionary computing could and should play an important role in the endeavor towards the Evolution of Things. However, the transition from digital and centralized evolutionary processes to physical and distributed evolution changes essential properties of the systems known and used in EC. This implies that evolutionary algorithms will have to be adjusted to cope with the new challenges. The resulting field could be seen as the 21st century incarnation of evolutionary computing with less emphasis on computing and more on evolutionary design, construction, and interaction with the environment. It can be expected that this field will benefit from certain algorithmic mechanisms in EC such that the wheel will not have to be reinvented. For instance, on-line evolutionary algorithms require on-line parameter setting mechanisms [28]. For some parameters, such as mutation rates or mutation step sizes, several methods are known in evolutionary computing and mechanisms for strongly embodied systems could be based on these.

In the following I discuss some problems raised by the Evolution of Things and show examples of existing work in EC that can be used to provide the first steps towards possible solutions.

Population Management. Population management may not be the most obvious problem raised by strongly embodied evolution, but it is literally a matter of life and death. In evolutionary computing the populations (almost) always have a fixed size, maintained by the centralized ‘manager’ that orchestrates the evolutionary operators. The essence of the mechanism is that survivor selection (a.k.a. replacement) is synchronized with reproduction in such a way that adding n new individuals is only possible if n old ones are removed. Likewise, n existing individuals are never removed without adding n new ones. This is certainly not the case in natural evolution. In general, situated evolution without central orchestration will rely on local, non-synchronized decisions regarding birth and death [36]. Hence, the existing individuals can be removed without adding new ones and new ones can be added without discarding old ones first. This implies that populations can shrink or grow. In extreme cases this can lead to complete extinction or overpopulation such that the evolutionary process is halted. Related work in

[29,43] addresses this issue by introducing *autonomous selection* of would-be parents as well as individuals targeted for termination in decentralized evolutionary algorithms. The mechanism has the following main features:

- Locally available global information. In particular, statistical information about the population's fitness (e.g. average fitness, min/max fitness) is available at each individual via a gossiping protocol.
- A locally executable function that determines selection probabilities for the given individual based on its own fitness and the available global information.
- An adaptation method that is regulating the parameters of the selection mechanism in each individual on-the-fly, depending on the course of the search.

Experiments demonstrate the feasibility of a fully decentralized evolutionary algorithm in which the population size can be kept stable. It is shown that parent and survivor selection can be done without central control, completely autonomously and asynchronously by the individuals themselves, yet avoiding the risk of population explosion or implosion.

The experiments cited above are carried out in traditional EA applications aiming at optimizing a given fitness function. In [22] the issue of possibly exploding or imploding populations is investigated in a more natural setting, in an ALife system where evolving agents decide autonomously and asynchronously if/when they reproduce. This is called *natural reproduction* and it is complemented by natural selection where an agent dies if it runs out of energy. The primary focus of the paper is the effect of adding individual learning (reinforcement learning) to the evolutionary mechanism with a learnable individual preference for performing the mating action. This allows for runtime control of reproduction rates and in principle it can optimally regulate population sizes. However, this also implies the possibility of unlearning mating and this is exactly what happened in the naive versions of the system, because reproduction offers no individual benefits but it does imply costs (children are expensive). Experiments showed that behavior optimal on individual level can have catastrophic effects on population level, leading to complete extinction. The paper also demonstrated that this effect can be counteracted by introducing a specific reward for the mating action that gives positive feedback to the agents, regardless the related costs. One could argue that this trick is just a reinvention of a solution known in nature, commonly called an orgasm. The system with such a special mating reward proved to be viable, although the right level of reward remained an open research question.

Twofold Fitness. The real world embedding in strongly embodied evolution mandates that the population is viable, i.e., can operate in the given environment that may be unknown beforehand and/or changing over time. In the meanwhile, most man-made systems are meant to serve a purpose, i.e., be useful for their designers/users. This implies that evolution should be employed for two purposes. Firstly, to provide a force for adaptation to the environment as it does in nature and in many artificial life implementations. This allows the evolving population to survive. Secondly, to provide a force for optimization towards the objectives set by the user as in mainstream evolutionary computing and evolutionary robotics. Recent work in [23,24,25] offers an algorithmic framework

to combine the drives for viability and utility. The **Multi-Objective aNd open-Ended Evolution** method (MONEE) balances evolution between environment-driven adaptation and task-driven optimization. It is based on the fact that evolutionary methods have two basic selection mechanisms and uses these in different roles: survivor selection is purely driven by the environment and parent selection is based on some user defined measure of task-performance. Experiments with large swarms of (simulated) e-pucks prove that MONEE does indeed promote task-driven behavior without compromising environmental adaptation. Furthermore, it is shown that an additional market mechanism can ensure equitable distribution of effort over multiple tasks.

6 In Vivo Veritas

The central thesis of this paper is that the Evolution of Things combines the controllability and programmability of artificial evolutionary systems as used in evolutionary computing and the physical embedding of natural evolutionary systems as seen in the biosphere. The corresponding research area will be concerned with populations of animate physical objects that undergo evolution in real space and real time driven by the environment, user preferences (if applicable), and their own decisions. Thus, as noted in Section 2, such artefacts can be perceived as robots in the broad sense with evolvable minds and bodies.

The emerging field can be seen as a synergetic combination of Evolutionary Computing, Robotics, Artificial Life, and Embodied AI. It will bring great new opportunities and imply great new challenges. Certainly, the EC community knows much about how to design, use, and analyze artificial evolutionary processes, but the whole body of work on ‘taming evolution’ in computer simulations may prove just a frivolous exercise. The examples reviewed in the previous section suggest that some of the existing EC techniques could be useful in the new setting, but in fact it is impossible to verify this without trying them. Real (world) problems will call for real (world) solutions.

To consider another angle let us recall the biological relevance discussed in Section 3. From this perspective strongly embodied evolutionary systems represent physical, rather than computational, models of evolution and this makes them more suited for biologically motivated studies. Such systems may not be based on the same biochemical micro-mechanisms as the carbon-based life on Earth, but they use the same macro-mechanisms (selection and reproduction with heredity) and *they are physically plausible*. This means that experimental findings will reveal much more about the real world than pure computer simulations.

7 Concluding Remarks

In this paper I argue that the science and technology of artificial evolution is on the verge of a major transition: from digital to physical, from software to hardware. I believe that within a few years we will have the technology for physically reproducing artefacts. Such artefacts may be ‘mindless’ or ‘animate’ and although evolving populations of ‘mindless’ passive artefacts will also be a novelty, the most interesting case is that of

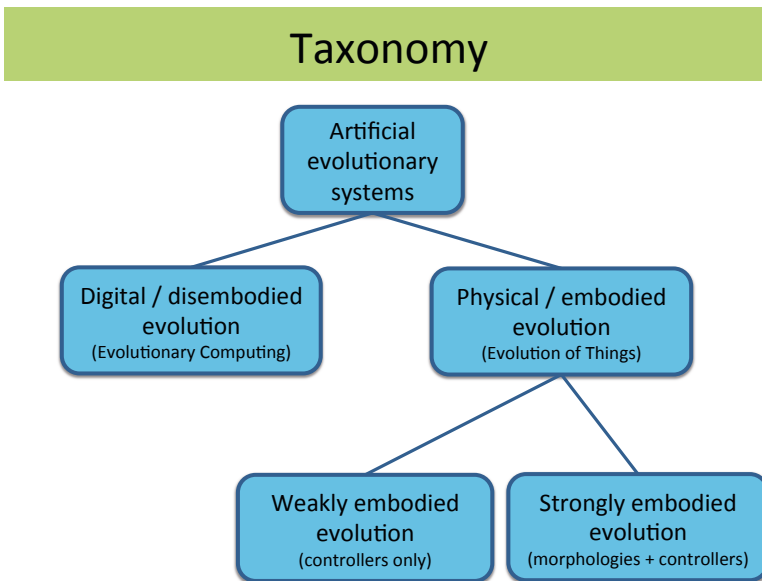


Fig. 8. Taxonomy of artificial evolutionary systems

autonomous animate artefacts capable of sensing, decision making, and performing actions on their own. Such entities—robots in the broad sense of the word, not necessarily based on a traditional mechatronic substrate— will be able to actively induce an evolutionary process ‘from within’—without a central evolutionary agency—in real time and real space. This new incarnation of artificial evolution will be a complete game changer confronting the designers of evolutionary mechanisms with unprecedented challenges.

Am I saying that Evolutionary Computing as we know it is doomed to disappear? Certainly not. Employing evolutionary algorithms for solving complex optimization and design problems is here to stay. The evolutionary algorithms used in these domains will become a subcategory of the bigger class of artificial evolutionary processes, that of *disembodied / digital* evolution, and I believe that this subcategory will remain relevant. However, I foresee that the Next Big Thing will be the emergence of *embodied / physical* artificial evolutionary systems, cf. Figure 8. Weakly embodied evolution will work on fixed hardware, such as populations of smart devices and/or robots that collectively evolve their control software without changing their physical makeup. Strongly embodied evolution (The Evolution of Things) will concern systems where the physical bodies co-evolve with the controllers. This will form a challenging area where I hope for exciting developments in the years to come.

Acknowledgments. I would like to thank my (former) colleagues and co-authors, and several fellow researchers abroad for the inspiring discussions that helped shape my views on this subject. In particular, I am indebted to Nicolas Bredeche, Evert Haasdijk, and Alan Winfield.

References

1. Anderson, M.: Embodied cognition: A field guide. *Artificial Intelligence* (149), 91–130 (2003)
2. Ashlock, D.: *Evolutionary Computation for Modeling and Optimization*. Springer (2006)
3. Auerbach, J., Bongard, J.: Environmental influence on the evolution of morphological complexity in machines. *PLOS Computational Biology* 10(1), e1003399 (2014)
4. Bentley, P. (ed.): *Evolutionary Design by Computers*. Morgan Kaufmann, San Francisco (1999)
5. Bentley, P., Corne, D.: *Creative Evolutionary Systems*. Morgan Kaufmann, San Francisco (2002)
6. Bongard, J.: Evolutionary robotics. *Communications of the ACM* 56(8), 74–85 (2013)
7. Brooks, R.: *Cambrian Intelligence: The Early History of the New AI*. MIT Press (1999)
8. Buresch, T., Eiben, A.E., Nitschke, G., Schut, M.: Effects of evolutionary and lifetime learning on minds and bodies in an artificial society. In: *Proceedings of the IEEE Conference on Evolutionary Computation, CEC 2005*, pp. 1448–1454. IEEE Press (2005)
9. Darwin, C.: *The Origin of Species*. John Murray, London (1859)
10. De Jong, K.: *Evolutionary Computation: A Unified Approach*. The MIT Press (2006)
11. De Jong, K.: Are genetic algorithms function optimizers? In: Männer, R., Manderick, B. (eds.) *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*, pp. 3–13. North-Holland, Amsterdam (1992)
12. Doncieux, S., Bredèche, N., Mouret, J.-B. (eds.): *New Horizons in Evolutionary Robotics*. SCI, vol. 341. Springer, Heidelberg (2011)
13. Eiben, A.: Grand challenges for evolutionary robotics. *Frontiers in Robotics and AI* 1(4) (2014), doi:10.3389/frobt.2014.00004
14. Eiben, A., Bredeche, N., Hoogendoorn, M., Stradner, J., Timmis, J., Tyrrell, A., Winfield, A.: The triangle of life: Evolving robots in real-time and real-space. In: Liò, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (eds.) *Advances in Artificial Life, ECAL 2013*, pp. 1056–1063. MIT Press (2013)
15. Eiben, A.E.: Multiparent recombination in evolutionary computing. In: Ghosh, A., Tsutsui, S. (eds.) *Advances in Evolutionary Computing*. Natural Computing Series, pp. 175–192. Springer (2002)
16. Eiben, A.E., Haasdijk, E., Bredeche, N.: Embodied, on-line, on-board evolution for autonomous robotics. In: Levi, P., Kernbach, S. (eds.) *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, ch. 5.2, pp. 361–382. Springer (May 2010)
17. Eiben, A.E., Kernbach, S., Haasdijk, E.: Embodied artificial evolution – artificial evolutionary systems in the 21st century. *Evolutionary Intelligence* 5(4), 261–272 (2012)
18. Eiben, A.E., Smith, J.: *Introduction to Evolutionary Computing*. Springer, Heidelberg (2003)
19. Fernando, C., Kampis, G., Szathmáry, E.: Evolvability of natural and artificial systems. *Procedia Computer Science* 7, 73–76 (2011)
20. Floreano, D., Husbands, P., Nolfi, S.: Evolutionary robotics. In: Siciliano, B., Khatib, O. (eds.) *Springer Handbook of Robotics*, vol. Part G.61, pp. 1423–1451. Springer (2008)
21. Floreano, D., Keller, L.: Evolution of adaptive behavior in robots by means of darwinian selection. *PLOS Biology* 8(1), e1000292 (2010)
22. Griffioen, A., Smit, S.K., Eiben, A.E.: Learning benefits evolution if sex gives pleasure. In: Michalewicz, Z., Reynolds, B. (eds.) *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, Hong Kong, China, pp. 2073–2080. IEEE Computational Intelligence Society. IEEE Press (2008)
23. Haasdijk, E., Bredeche, N.: Controlling task distribution in MONEE. In: Liò, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (eds.) *Advances in Artificial Life, ECAL 2013*, pp. 671–678. MIT Press (2013)

24. Haasdijk, E., Bredeche, N., Eiben, A.E.: Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PLoS ONE* 9(6), e98466 (2014)
25. Haasdijk, E., Weel, B., Eiben, A.: Right on the MONEE. In: Blum, C., et al. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2013)*, Amsterdam, The Netherlands, July 6-10, pp. 207–214. ACM (2013)
26. Hoffmann, M., Pfeifer, R.: The implications of embodiment for behavior and cognition: animal and robotic case studies. In: Tschacher, W., Bergomi, C. (eds.) *The Implications of Embodiment: Cognition and Communication*, pp. 31–58. Imprint Academic (2012)
27. Iida, F., Pfeifer, R., Steels, L., Kuniyoshi, Y. (eds.): *Embodied Artificial Intelligence*. LNCS (LNAI), vol. 3139. Springer, Heidelberg (2004)
28. Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation* (to appear, 2014), doi:10.1109/TEVC.2014.2308294
29. Laredo, J., Eiben, A.E., van Steen, M., Merelo, J.J.: EvAg: a scalable peer-to-peer evolutionary algorithm. *Genetic Programming and Evolvable Machines* 11, 227–246 (2010)
30. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. *Nature* 406, 974–978 (2000)
31. Long, J.: *Darwin's Devices: What Evolving Robots Can Teach Us About the History of Life and the Future of Technology*. Basic Books (2012)
32. Nelson, A.L., Barlow, G.J., Doitsidis, L.: Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems* 57(4), 345–370 (2009)
33. Nolfi, S., Floreano, D.: *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge (2000)
34. Pfeifer, R., Bongard, J.: *How the Body Shapes the Way We Think*. MIT Press (2006)
35. Pollack, J.B., Lipson, H., Ficici, S.G., Funes, P., Hornby, G., Watson, R.A.: Evolutionary techniques in physical robotics. In: Miller, J.F., Thompson, A., Thompson, P., Fogarty, T.C. (eds.) *ICES 2000*. LNCS, vol. 1801, pp. 175–186. Springer, Heidelberg (2000)
36. Schut, M., Haasdijk, E., Eiben, A.E.: What is situated evolution? In: *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May 18-21, pp. 3277–3284. IEEE Press, Piscataway (2009)
37. Sims, K.: Evolving 3D morphology and behavior by competition. *Artificial Life* 1(4), 353–372 (1994)
38. Trianni, V.: *Evolutionary Swarm Robotics – Evolving Self-Organising behaviors in Groups of Autonomous Robots*. SCI, vol. 108. Springer, Heidelberg (2008)
39. Vargas, P., Paolo, E.D., Harvey, I., Husbands, P. (eds.): *The Horizons of Evolutionary Robotics*. MIT Press (2014)
40. Waibel, M., Floreano, D., Keller, L.: A quantitative test of Hamilton's rule for the evolution of altruism. *PLoS Biology* 9(5), e1000615 (2011)
41. Wang, L., Tan, K., Chew, C.: *Evolutionary Robotics: from Algorithms to Implementations*. World Scientific Series in Robotics and Intelligent Systems, vol. 28. World Scientific (2006)
42. Watson, R.A., Ficici, S.G., Pollack, J.B.: Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems* 39(1), 1–18 (2002)
43. Wickramasinghe, W., van Steen, M., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: D. T., et al. (eds.) *GECCO 2007: Proc of the 9th Conference on Genetic and Evolutionary Computation*, pp. 1460–1467. ACM Press (2007)

Online Black-Box Algorithm Portfolios for Continuous Optimization

Petr Baudiš and Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Technická 2, 166 27 Prague 6, Czech Republic
pasky@ucw.cz, petr.posik@fel.cvut.cz

Abstract. In black-box function optimization, we can choose from a wide variety of heuristic algorithms that are suited to different functions and computation budgets. Given a particular function to be optimized, the problem we consider in this paper is how to select the appropriate algorithm. In general, this problem is studied in the field of algorithm portfolios; we treat the algorithms as black boxes themselves and consider *online* selection (without learning mapping from problem features to best algorithms a priori and dynamically switching between algorithms during the optimization run).

We study some approaches to algorithm selection and present two original selection strategies based on the UCB1 multi-armed bandit policy applied to unbounded rewards. We benchmark our strategies on the BBOB workshop reference functions and demonstrate that algorithm portfolios are beneficial in practice even with some fairly simple strategies, though choosing a good strategy is important.

1 Introduction

Continuous black-box optimization concerns itself with the problem of finding a minimum value of a real-parameter function that has inaccessible analytical form. This is a rich area of research that produced many algorithms over the last 50 years — from the venerable Nelder-Mead simplex algorithm [1] to various gradient descent methods to population-based methods.

However, if a function is truly “black-box” and its features are hard to predict, the key question in the face of such variety is “which algorithm should I choose?” We can turn for help at a common platform for performance comparison — the currently accepted de-facto standard is the *COmparing Continuous Optimisers* COCO platform [2] [3] that was originally developed for the BBOB workshop series and which provides (most importantly) a set of diverse reference benchmark functions. But there is still a long way from previously published performance results on reference functions to a decision about which algorithm to use on an arbitrary function provided by the user. A method to automate the process is certainly desirable.

The problem of algorithm selection is not new [4] and was so far popular mainly when applied to combinatorial problem solvers [5]. In our work, we adopt

the prism of algorithm portfolios [6]. Let us have a set of *heuristic algorithms* (each suitable for a different class of problem instances). Given a problem instance, we apply a *selection strategy* to pick and apply an algorithm from this portfolio. We can perform the selection once or along a fixed schedule based on features of the problem instance (*offline selection*), or in multiple rounds allocating time to portfolio members based on their performance in previous rounds (*online selection*). In successive rounds, algorithms can be either resumed from their previous state or restarted; we take the approach of resuming them, reserving the restart schedule to be an internal matter of each algorithm.

Using algorithm portfolios for continuous black-box optimization is still a fresh area of research. The main results so far lie either in modifications of population methods that combine a variety of genetic algorithms together, e.g. the *MultiEA* [7], *AMALGAM-SO* [8] and *PAP* [9] methods; or in offline methods based on exploratory landscape analysis [10] which were also recently applied to the BBOB workshop scenario [11]. We could also draw ideas from hyper-heuristics and adaptive strategies for operator selection within population-based algorithms [12] [13]. A Multi-Armed Bandit scenario has been already considered in the field of function optimization for determining online restart schedules [14].

In our work, we enforce the distinction between algorithms and selection strategies — we regard the algorithms as *black-box*, completely avoiding any modification and simply repeatedly resuming them and allowing them to make another optimization step. This allows any already implemented state-of-art algorithms to be easily combined in a single portfolio and also extends to other than population-based methods. Furthermore, we focus on *online* adaptive selection that selects algorithms based on their performance so far and does not require possibly expensive or brittle feature extraction and training.

We aim to confirm whether using an algorithm portfolio is advantageous compared to investing the whole budget in a single overallly dominant algorithm, and how do selection strategies influence the portfolio performance.

In section 2, we investigate algorithm selection through the paradigm of the Multi-Armed Bandit Problem, motivating the proposed selection strategies. In section 3, we describe the exact strategies we have compared in our experiments. We present the experimental results in section 4 and draw our conclusions and outline some directions for future work in section 5.

2 Algorithm Selection as Multi-armed Bandit Problem

When considering an action selection strategy that operates in an initially unknown environment, we face the fundamental dilemma of balancing exploitative actions maximizing the reward based on our current model of the environment and exploratory actions that refine the model. Multi-Armed Bandit Problem [15] is a reference statistical problem that allows abstract study of the exploration-exploitation dilemma.

2.1 Multi-armed Bandit Problem

In its typical formulation [16], a K -armed bandit problem is defined by a sequence of random rewards $X_{it} \in [0, 1]$, $i = 1, \dots, K$, $t \in \mathbb{N}$, where i is an index of the arm of a bandit (in other words, a gambling machine) and t denotes successive pulls of the arm. All rewards are independent random variables and rewards of a single arm follow an identical stationary distribution, but the distribution and expected value are originally unknown. A bandit policy π is then a function that selects the next arm to be pulled based on the sequence of rewards up to that point. The goal is to maximize cumulative reward over time; the policy aims to pull the arm with highest expected reward (exploitation), but needs to continually update its belief about which arm has the best reward (exploration).

The measure of policy performance is its “regret”, i.e. cumulative reward loss compared to a hypothetical oracle policy. Let μ_i denote the true expected reward of arm i , μ^* the true expected reward of the optimal arm, $T_i(n)$ the number of times arm i has been pulled up to the n -th pull, and R_n the current regret:

$$R_n = n\mu^* - \sum_{i=1}^K \mathbb{E}[T_i(n)] \mu_i$$

It was proven early [16] that the lower asymptotic bound for the regret R_n is $\Omega(\ln n)$, and many (even very simple) policies achieve this bound.

Perhaps the simplest policy is the **epsilon-greedy policy**, simply choosing the arm with the highest estimated expectation with probability $1 - \varepsilon$ and a uniformly random arm with probability ε . Therefore, the ratio of exploration and exploitation actions is fixed and uniform exploration strategy is applied.

The **Upper Confidence Bound (UCB1) policy** [17] implicitly negotiates the exploration-exploitation dilemma by adding a relative measure of uncertainty (*bias*) to the estimated expectation; therefore, even low-expectation arms are occasionally explored when the uncertainty is too high compared to other arms:

$$\pi_{\text{UCB1}}(n) = \operatorname{argmax}_i \left(\hat{\mu}_i(n) + c \sqrt{\frac{2 \ln n}{T_i(n)}} \right) \quad (1)$$

The policy quickly gained popularity as a reference Multi-Armed Bandit policy since it can be proven that the policy follows the logarithmic regret bound not just asymptotically but also uniformly (after a burn-in period) if the parameter c is tuned for the optimal exploration-exploitation ratio.

2.2 Action Rewards versus Optimization Performance

To apply the Multi-Armed Bandit Problem on algorithm selection, we (analogously to e.g. [14]) represent each algorithm as an arm and in each algorithm iteration decide which arm(s) to step once next. However, the key question is how to represent the reward estimates¹ used for the decision.

¹ In some of the algorithm selection literature, this is termed “credit” of the algorithm.

In the simplest form, the reward estimate $\hat{\mu}$ may be represented simply by the negative of the **raw value** of the function in the current iteration of the algorithm — therefore, the algorithm currently closest to the optimum will be associated with the highest reward estimate.

This approach may be problematic if the reward needs to be bounded in a fixed interval; a normalization strategy is proposed below in Sec. 2.3. However, when the value does not approach the optimum smoothly, absolute value difference may not correspond well to algorithm performance difference. The approach of **value rank** [13] sidesteps the issue by ranking algorithms based on the values they yield in each round and using that rank (normalized by linear rescaling to $[0, 1]$) as the reward estimate.

An extension of these approaches is, instead of considering just the latest normalized reward, to use an exponentially decaying average of recent normalized rewards with an adaptation rate α [12], also known as the **exponentially weighted moving average (EWMA)**.

The Multi-Armed Bandit Problem assumes that the reward distributions are stationary and rewards are independent. But this is clearly not the case in our setting — as each algorithm proceeds through the functional landscape, its rate of improvement changes and previous results are tied to its future performance. These assumption violations may not be fatal in practice and we test the performance of considered algorithms without regard to them. Furthermore, it has been proven that the UCB1 policy can be used as-is for non-stationary distributions [18] (provided that the c parameter has been set correctly).

2.3 Raw Values and the UCB1 Policy

The UCB1 policy sums the reward estimate $\hat{\mu}$ with a bias term (multiplied by a fixed constant). A key assumption here is that $\mu \in [0, 1]$ (being a reward expectation), but our raw values are entirely unbounded and exponentially skewed. A simple work-around is to use the value rank instead, but the actual difference between values may be useful during the decision, therefore we also propose a raw value normalization approach.

Every time we invoke the UCB1 policy, we re-normalize values of all arms, taking two assumptions. First, we use values relative to the *supposed optimum* by always assuming that we are just short of it, i.e. putting the $f_{opt?} = \min f - \Delta f$ where $\Delta f = 10^{-8}$ is the target precision of COCO. Secondly, we assume that the algorithms converge exponentially fast², therefore differences (relative to the supposed optimum $f_{opt?}$) between 10^3 and 10^2 should be considered on the same scale as differences between 10^{-6} and 10^{-7} .

With these two assumptions, a *log-rescaling* process is straightforward. First, we convert the absolute f_i values to values relative to the supposed optimum and rescale the values logarithmically:

$$g_i = \log(f_i - f_{opt?})$$

² A similar idea appears within the MetaMax algorithm [14] and is also supported by practical observations.

Second, we assign rewards by linear rescaling of the preprocessed values:

$$\mu_i = 1 - \frac{g_i - \min_j g_j}{\max_j g_j - \min_j g_j}$$

3 Algorithm Selection Strategies

The strategies and reward schemes outlined above offer a wide variety of possible combinations. To focus the scope of our research, we considered only combinations already proposed in the literature, in addition to a baseline strategy and two new applications of the UCB1 policy we propose. We performed a rough parameter tuning of each of the considered strategies on a portfolio of seven algorithms (see Sec. 4 and [19]); the performance is not very sensitive to exact values of the parameters. Our main loop consist of selecting an algorithm to step, then running it for a single iteration, then repeating the selection etc.

RR: As one baseline strategy, we used a round robin policy that samples each algorithm equally, in their portfolio order. (This is different from a “run in parallel” strategy in that the algorithms consume different budgets to sample a single iteration.)

EG: The epsilon-greedy policy with $\varepsilon = 0.5$.

RUCB: The UCB1 policy with EWMA-recent ranks as reward estimates ($\hat{\mu}_i$ in Equation 1), with $c = 8$ and adaptation rate $\alpha = 0.9$.

LUCB: The UCB1 policy with EWMA-recent log-rescaled values as reward estimates ($\hat{\mu}_i$ in Equation 1), with $c = 16$ and adaptation rate $\alpha = 0.7$.

We also tested Probability Matching and Adaptive Pursuit [12], Threshold Ascent [20], MetaMax variants [14] and UCB1 with Sum-of-Ranks and Area-Under-the-Curve rewards [13]. However, their results were not competitive with the strategies above and we cannot elaborate on them due to space limitations³.

4 Experiments and Results

To benchmark against the BBOB testbed, we used the reference COCO framework [2] [3]. Our “COCOpf” extension [19] provides a common algorithm portfolio codebase, including algorithm stepping and publication reports generation.

We use the reference portfolio of seven optimization algorithms — the CMA-ES algorithm [21] and six numerical optimization algorithms distributed along the SciPy software package [22]. Description and comparison of the individual algorithms is detailed in [19].

Within the COCO framework, functions are classified based on their properties to separable, multi-modal, etc. Here, we deliberately did not adopt this classification as we do not study the behavior of individual algorithms⁴, but the behavior of strategies that depend on the performance of portfolio members.

³ Their results are included in an extended version of this paper, the raw datasets and generated reports at <http://pasky.or.cz/sci/cocopf-opt13>.

⁴ See [19] for plots of performance of the portfolio members in the reference COCO function classes.

Table 1. The assignment of individual COCO benchmark functions to the classes we have devised, determined on the performance of our portfolio on the functions. Vertical lines delineate the standard function classes used by COCO.

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
By solvers (s/m)	m	s	s	s	m	s	s	m	m	s	s	s	s	m	s	s	s	s	s	s	s	s	s	s	m
By winner (g/b)	b	b	b	b	b	g	g	b	b	g	g	g	g	b	b	g	g	g	b	b	g	g	g	g	b
By converg. (s/v)	s	s	s	s	s	v	s	v	v	v	v	v	v	v	v	s	s	s	v	v	v	v	v	v	v

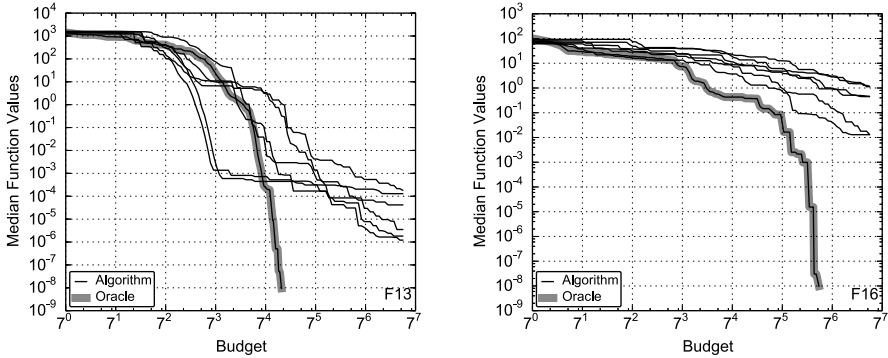


Fig. 1. Typical volatile (left) and stable (right) portfolio convergence. (We distinguish only the portfolio’s best algorithm for each considered function.)

With respect to a particular portfolio, we propose the following function classification:

- **By solvers** based on whether a single algorithm dominated others in performance (*single-solver*), or if multiple algorithms converged similarly quickly (*multi-solver*). The latter favors strategies that do not focus sharply on a single algorithm. We consider algorithms to perform similarly if one takes at most twice as many evaluations to reach the optimum than another.
- **By winner** as the clear overall winning algorithm in our portfolio is CMA, converging first on most functions (*CMA-good*), but there are many functions on which CMA actually performs relatively very poorly (*CMA-bad*). Splitting functions along these lines allows us to quantify the loss caused by using a portfolio instead of a single algorithm.
- **By convergence** based on the algorithm behavior before finding optimum. In some cases, the algorithms that yield the best function values early continue to dominate throughout the convergence progression and eventually indeed converge first — these are the *stable* functions where strategies can early focus on the best algorithms. On the other hand, especially in a landscape rich on difficult-to-escape local optima, the pace of convergence of conventional algorithms stays the same or even slows down throughout their run, while an algorithm (often CMA) that produced lukewarm results with smaller

Table 2. The average rank of portfolio algorithms and strategies (computed over both) by the order in which they converge

Solver	all	multi	single	volatile	stable	CMA-good	CMA-bad
CMA	6.4	11.2	4.8	4.7	9.2	1.3	11.4
BFGS	9.4	4.7	11.0	7.5	12.7	12.2	6.6
L-BFGS-B	9.7	3.8	11.7	8.1	12.5	13.0	6.5
SLSQP	10.9	6.0	12.6	9.5	13.2	13.5	8.3
N.-M.	12.0	10.0	12.7	9.9	15.6	12.0	12.0
Powell	12.9	12.9	12.9	15.9	7.9	14.6	11.3
CG	13.3	6.6	15.6	13.0	13.8	15.8	10.8
LUCB	6.5	10.5	5.1	7.2	5.3	4.4	8.5
RUCB	6.8	8.2	6.3	6.5	7.2	6.5	7.1
EG	7.1	7.2	7.1	8.3	5.2	6.6	7.6
RR	9.3	11.0	8.7	9.1	9.7	8.2	10.4

budget suddenly and unexpectedly improves in a dramatic way, achieving convergence early after⁵ — we term these *volatile* functions. These are obviously much harder for purely online strategies. We consider a function to be volatile if the algorithm that converges first in budget $|\text{portfolio}|^k$ for some k was not the best algorithm in budget $|\text{portfolio}|^{k-2}$.

We illustrate the typical convergence progression on volatile vs. stable functions in figure 1⁶. Table 1 shows the classification of functions for our portfolio.

In all results, we show measurements with maximum function evaluation budget $\text{dim} \cdot 10^5$ and use the performance on 5D functions as at least one algorithm converges within our budget for almost all functions in this dimension.

Table 2 shows the *average final rank* of each algorithm and portfolio strategy (in terms of budget required for convergence, i.e. 1 is best) averaged over the individual classes. Table 3 shows the *average⁷ strategy log-slowdowns* in terms of difference between budget b required for certain algorithm or strategy to converge and budget b_o required for the oracle strategy (which runs only the single best algorithm for each function) to converge, computed as $\log_{|\text{portfolio}|}(b/b_o)$. I.e. a slow-down of 0 means perfect performance and slow-down of 1 means that it is as if we simply run all algorithms in parallel⁸.

We can observe that the LUCB strategy performs best; while in total average it is superseded by the winner algorithm CMA, that is not very surprising as this algorithm performs best on its own on half of the functions and portfolios will always introduce an overhead. At the same time, the LUCB strategy exhibits

⁵ We did not observe a situation where the initially best algorithm temporarily loses its first place only to eventually converge first.

⁶ Interested readers may find portfolio convergence graphs for all functions as well as raw datasets at <http://pasky.or.cz/sci/cocopf-opt13>.

⁷ Within a single function, the median instance is considered. Across functions within a class, the slowdown is averaged.

⁸ Functions on which no algorithm converges in the assigned budget are not included in the average. We assign a log-slowdown of 3 to strategies not converging in time.

Table 3. The average log-slowdown of portfolio algorithms and strategies compared to oracle strategy (i.e. the best algorithm for each function)

Solver	all	multi	single	volatile	stable	CMA-good	CMA-bad
CMA	0.7	1.1	0.6	0.5	1.1	0.0	1.5
CG	2.3	0.8	2.7	2.3	2.3	2.8	1.7
BFGS	1.5	0.7	1.8	1.4	1.7	2.2	0.8
L-BFGS-B	1.9	0.6	2.3	1.8	1.9	2.5	1.1
Nelder-Mead	2.1	0.8	2.6	2.0	2.4	2.5	1.8
SLSQP	2.3	1.0	2.8	2.4	2.3	2.9	1.8
Powell	2.3	2.0	2.4	3.0	1.2	3.0	1.6
LUCB	0.9	0.9	0.9	1.3	0.3	1.1	0.8
RUCB	1.3	0.9	1.4	1.4	1.1	1.7	0.8
EG	1.4	1.0	1.6	1.6	1.1	2.0	0.9
RR	1.5	1.1	1.7	1.8	1.2	2.1	1.0

less performance variation (in terms of log-slowdown) from class to class, and on stable functions it outperforms all other strategies *and* algorithms by a large margin. The RUCB and EG strategies can also outdo the LUCB strategy on function-by-function basis in some classes, as is apparent from the average ranks.

5 Discussion and Conclusion

The results demonstrate a good case for the usage of algorithm portfolios for black-box optimization. Overall, our proposed LUCB and RUCB strategies perform the best, but the very simple EG strategy also performed very well. We believe both the LUCB and EG strategies are easy to implement and can be used as reference strategies in further research.

As expected, the portfolios were very beneficial especially in case of non-volatile functions. More work is clearly needed to deal with volatile functions. That will further benefit even non-volatile performance as all our strategies are currently very explorative — investment in even bad-looking algorithms is important in cases of volatile functions. Regardless of function classes, algorithm portfolios also offer a more stable performance than an individual algorithm in the face of unknown (as even CMA can fail miserably on some of the functions).

We can observe a significant stratification among the tested selection strategies. We can conclude that a selection strategy matters, and even a simple strategy like EG will bring a big improvement over a round-robin selection strategy which is still the de-facto standard for algorithm portfolios when they are used.

5.1 Future Work

We hope that performance can be improved by a future portfolio that is more diverse and balanced (either thanks to more algorithms or parameter tuning to

refocus different algorithms to specific function classes). Another step in this direction is to study the influence of portfolio size and composition on performance of various strategies⁹.

Clearer measures for adaptation lag when the best algorithm changes in volatile functions or the suitability of using function value differences in strategies would be very desirable.

Many approaches to algorithm selection in terms of Multi-Armed Bandit Policies and reward assignment were proposed in the literature. We could not consider them all, but we think that especially performance-modeling approaches like modifications of the MultiEA [7] or GambleTA [23] algorithms are worth investigating in the future.

Aside of that, we attempted to give the problem a modular structure; this allows e.g. a full-scale comparison of reward assignment and bandit policy combinations on top of what has been proposed in the literature so far. Furthermore, the usage of UCB1 is not theoretically very sound and we assume it should be possible to develop a more suitable policy formula.

We have investigated just a purely online, black-box mode of action so far, but there is certainly a room to grow in the direction of previously introduced approaches. Offline learning can be combined with online methods at least to initialize them or detect function classes. Intermediate solutions could be shared and migrated between individual algorithms.

Acknowledgements. This research was supported by the CTU grant SGS14/194/OHK3/3T/13 “Automatic adaptation of search algorithms”. We would like to thank the Department of Applied Mathematics of the Charles University in Prague for providing the computational resources that made our experiments possible. Matteo Gagliolo’s introduction to the research landscape helped us much in the beginning of our work.

References

1. Nelder, J.A., Mead, R.: A simplex method for function minimization. *The Computer Journal* 7(4), 308–313 (1965)
2. Hansen, N., et al.: Comparing continuous optimisers: Coko, <http://coco.gforge.inria.fr/>
3. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA (2012)
4. Rice, J.R.: The algorithm selection problem. *Advances in Computers* 15, 65–118 (1976)
5. Kotthoff, L.: Algorithm selection for combinatorial search problems: A survey. *AI Magazine* (2014)
6. Gomes, C.P., Selman, B.: Algorithm portfolios. *Artificial Intelligence* 126(1), 43–62 (2001)

⁹ For example, it surprised us that it seems trimming the current portfolio by worst performing algorithms does not improve overall performance. However, we still consider this result preliminary.

7. Yuen, S.Y., Chow, C.K., Zhang, X.: Which algorithm should i choose at any point of the search: An evolutionary portfolio approach. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. GECCO 2013, pp. 567–574. ACM, New York (2013)
8. Vrugt, J.A., Robinson, B.A., Hyman, J.M.: Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Trans. on Evolutionary Computation* 13(2), 243–259 (2009)
9. Peng, F., Tang, K., Chen, G., Yao, X.: Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation* 14(5), 782–800 (2010)
10. Muñoz, M.A., Kirley, M., Halgamuge, S.K.: The algorithm selection problem on the continuous optimization domain. In: Moewes, C., Nürnberger, A. (eds.) *Computational Intelligence in Intelligent Data Analysis*. SCI, vol. 445, pp. 75–89. Springer, Heidelberg (2013)
11. Bischl, B., Mersmann, O., Trautmann, H., Preuss, M.: Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference, pp. 313–320. ACM (2012)
12. Thierens, D.: Adaptive strategies for operator allocation. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) *Parameter Setting in Evolutionary Algorithms*. SCI, vol. 54, pp. 77–90. Springer, Heidelberg (2007)
13. Fialho, A., Schoenauer, M., Sebag, M.: Toward comparison-based adaptive operator selection. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 767–774. ACM (2010)
14. György, A., Kocsis, L.: Efficient multi-start strategies for local search algorithms. *J. Artif. Int. Res.* 41(2), 407–444 (2011)
15. Robbins, H.: Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society* 58, 527–535 (1952)
16. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6, 4–22 (1985)
17. Auer, P., Bianchi, N.C., Fischer, P.: Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47(2/3), 235–256 (2002)
18. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) *ECML 2006*. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006)
19. Baudiš, P.: COCOpf: An algorithm portfolio framework. In: Poster 2014 — the 18th International Student Conference on Electrical Engineering, Czech Technical University, Prague, Czech Republic (2013)
20. Streeter, M.J., Smith, S.F.: A simple distribution-free approach to the max k -armed bandit problem. In: Benhamou, F. (ed.) *CP 2006*. LNCS, vol. 4204, pp. 560–574. Springer, Heidelberg (2006)
21. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., Larranaga, P., Inza, I., Bengoetxea, E. (eds.) *Towards a New Evolutionary Computation*. STUDEFUZZ, vol. 192, pp. 75–102. Springer, Heidelberg (2006)
22. Jones, E., Oliphant, T., Peterson, P., et al.: *SciPy: Open source scientific tools for Python* (2001)
23. Gagliolo, M., Schmidhuber, J.: Algorithm portfolio selection as a bandit problem with unbounded losses. *Annals of Mathematics and Artificial Intelligence* 61(2), 49–86 (2011)

Shuffle and Mate: A Dynamic Model for Spatially Structured Evolutionary Algorithms

Carlos M. Fernandes^{1,2}, Juan L.J. Laredo³, Juan Julian Merelo², Carlos Cotta⁴,
Rafael Nogueras⁴ and Agostinho C. Rosa¹

¹ LaSEEB-ISR-IST, University of Lisbon, Portugal

² Department of Architecture and Computer Technology, Univ. of Granada, Spain

³ Faculty of Sciences, Technology and Communications, Univ. of Luxembourg

⁴ Departamento de Lenguajes y Ciencias de la Computación, Univ. of Malaga, Spain
{cfernandes, acrosa}@laseeb.org, juan.jimenez@uni.lu,
{jjmerelo, rafael.nogueras}@gmail.com, ccottap@lcc.uma.es

Abstract. This paper studies a self-organized framework for modeling dynamic topologies in spatially structured Evolutionary Algorithms (EAs). The model consists of a 2-dimensional grid of nodes where the individuals interact and self-organize into clusters. During the search process, the individuals move through the grid, following a pre-defined simple rule. In order to evaluate the model, a dynamic cellular Genetic Algorithm (dcGA) is built over the proposed topology and four different movement rules are tested. The results show that when the ratio between the number of nodes in the grid and the population size is above 4:1, the individuals self-organize into highly dynamic clusters and significantly improve results attained by standard cGAs with static topologies on a set of deceptive and multimodal functions.

1 Introduction

In panmictic populations, every individual is allowed to interact with every other individual. Standard Evolutionary Algorithms (EAs) mimic this strategy for parent selection and recombination, but large-scale problems or deceptive functions with multiple local optima may require other type of population structures. In recent years, non-panmictic EAs [10], which restrict the interaction according to a pre-defined or evolving structure, are gaining increasing attention by the community.

In non-panmictic EAs, the population structure specifies a network of acquaintances over which individuals can interact (i.e., mating or selection is restricted to neighborhoods within the network). These non-panmictic EAs are also known as *spatially structured* EAs [10], a category that includes fine-grained approaches such as cellular EAs (cEAs) [1] and coarse-grained approaches such as island models [2]. In cEAs, the population is distributed in a grid and the interaction is restricted to the individuals' neighborhood. In island EAs, different subpopulations evolve isolated from each other and occasionally exchange individuals using a predefined strategy which specifies the rate and quantity of information to transfer.

The main disadvantage with island and cellular EAs is that their base-structures, which have a great influence on the algorithm performance, require extra designing and tuning effort. In the case of island models, this added complexity translates in deciding policies for the migration frequency, selection and replacement of migrants and the topology itself. As for traditional cEAs, they use static structures that impose a rigid connectivity between the individuals.

Furthermore, even though cEAs may achieve a better management of the genetic diversity in the population when compared to panmictic EAs, the balance between exploration and exploitation may not be sufficient for problems with deceptive or epistatic fitness landscapes. Since the population is globally connected, information may spread quickly and local optima can easily take over the entire population. The investigation in this paper is an attempt to design a simple dynamic topology for cEAs, with a varying neighborhood degree and an intrinsic clustering behavior that approaches the cEA to an island model, keeps genetic diversity at a higher level and prevents sub-optimal solutions to take over the population.

In the proposed topology, n individuals are distributed in a 2-dimensional m –nodes grid where it holds $m > n$. Every time-step, each individual tries to recombine with one of the individuals in its Moore neighborhood (if there are any). Furthermore, the structure is dynamic: in each time-step, every chromosome updates its position by moving to a neighboring node (if there are empty nodes in the individual’s neighborhood), according to a pre-defined rule that selects the destination. The position update rule, which is implemented locally and without any knowledge on the global state of the system, can be based on stigmergy [5] or Brownian movements.

When stigmergic behavior is induced by a stigmergic rule (i.e., individuals communicate via the environment), different niches of individuals appear and disappear at run-time. This clustering behavior is an emergent property of the model, and the resulting cEA has certain resemblance with an island model, with dynamic clusters (or sub-populations) of individuals with varying size.

We hypothesize that with this scheme the population diversity decreases at a lower rate (when compared to a standard topology), and, as a consequence, the performance of the cEA on deceptive and hard problems is improved. In this paper, the dynamic topology is tested on a cellular Genetic Algorithm (cGA). Four different strategies are described for the position update rule and the resulting algorithms are tested with a set of deceptive and epistatic functions that challenge EAs’ abilities to combine building-blocks. The results show that when the ratio between the number of nodes in the grid and population size is above 4:1, the dynamic cGAs converge more often to the global optimum and significantly improve the performance of standard cGAs.

The remaining of the paper is structured as follows: Section 2 gives a background review on cEAs and on dynamic alternative topologies for cEAs; Section 3 describes the proposed system; Section 4 describes the experiments and discusses the results; Section 5 concludes the paper and outlines future lines of work.

2 Background Review

The initial objective of spatially structured EAs was to develop a framework for studying massive parallelization. However, the need to provide traditional EAs with a

proper balance between exploration and exploitation and overcome standard EAs drawbacks, like synchronicity, rigid connectivity and strong dependence on the problem, motivated several lines of research that explore the potentiality of different population structures in maintaining genetic diversity [10]. Additionally, complex population structures have been studied, some of them under the knowledge provided by recent developments in network theory.

In [1], Alba and Dorronsoro dynamically change the ratio that defines the neighborhood of interaction. Since the ratio may affect selection pressure, the authors analyze the influence of its value on the balance between exploration and exploitation. However, the base-structure of the cellular EA is maintained throughout the run. In [11], Whitacre et al. focus on two important conditions missing in EA populations: a self-organized definition of locality and interaction epistasis. With that purpose in mind, they proposed a dynamic structure and concluded that the two features, when combined, provide behaviors not present in the traditional spatially structured EAs. The most noticeable change is an unprecedented capacity for sustainable coexistence of genetically distinct individuals within a single population. The authors state that the capacity for sustained genetic diversity is not imposed on the population; instead, it emerges as a natural consequence of the dynamics of the system. Laredo et al. [7] proposed a framework for EAs based on peer-to-peer networks [9]. Within a simulated network, they model the dynamics of real networks and conclude that their system is able to achieve better performance than traditional EAs on a wide range of problems, while being scalable and resilient to the volatility of nodes in the network.

In order to deal with the specific issues that may affect the design and performance of spatially structured EAs, Fernandes et al. [3] devised a complex adaptive system to be used as a dynamic structure for populations. This model, which can be regarded as a cellular automaton [6] with short-term memory, uses stigmergic communication and simple rules for movement on a grid of nodes, giving rise to self-organized clusters of particles. A noticeable feature of these clusters is that they keep evolving and changing shape, thus providing some kind of highly dynamic order. The authors demonstrate that the proposed system has indeed emergent properties that may prove useful for spatially structured EAs, or other spatially structured population-based metaheuristics.

In fact, this framework has been recently used to implement a spatially structured Particle Swarm Optimization (PSO) algorithm, in which the particles' interaction is defined by their position on the grid [4]. In this case, the position update rule is based on Brownian movement. Recently, Nogueras et al. [8] adapted the model in [3] to a spatially structured multimemetic algorithm with dynamic topology. The authors show that the dynamic topology maintains genetic diversity at a higher level and reduces the rate of convergence to local optima.

In this paper we take a different approach and test the framework as a dynamic topology for cellular Genetic Algorithms (cGAs), using position update rules that model Brownian movement and stigmergic behavior. The base model and the proposed topology are described in the following section.

3 Dynamic Topology

This section gives a formal description of the network and the transition rules that define the proposed model for dynamic population structures.

Let us consider a rectangular grid G of size $q \times s \geq \mu$. Each cell G_{uv} of the grid is a tuple $\langle \eta_{uv}, \zeta_{uv} \rangle$, where $\eta_{uv} \in \{1, \dots, \mu\} \cup \{\bullet\}$ and $\zeta_{uv} \in (D \times \mathbb{N}) \cup \{\bullet\}$, for some domain D . The value η_{uv} indicates the index of the individual that occupies the position $\langle u, v \rangle$ in the grid. If $\eta_{uv} = \bullet$ then the corresponding position is empty. However, that same position may still have information, namely a mark (or clue) ζ_{uv} , that is placed by the individuals and provides a form of communication between them. If $\zeta_{uv} = \bullet$ then the position is empty and unmarked. Please note that when $q \times s = \mu$, the topology is the standard static 2-dimensional structure.

The marks are placed by individuals that occupied that position in the past and they consist of information about those individuals (captured by domain D), like their fitness ζ_{uv}^f or a copy of their genotype \vec{x} , as well as a time stamp ζ_{uv}^t that indicates the iteration in which the mark was placed. The marks have a lifespan of K iterations, after which they are deleted.

Initially, $G_{uv} = (\bullet, \bullet)$ for all $\langle u, v \rangle$. Then, individuals are placed randomly on the grid (only one individual per node). Afterwards, all individuals are subject to a movement phase (or position update), followed by an evolutionary phase. The process (position update and evolutionary phase) repeats until a stop criterion is met.

The evolutionary phase is the standard iteration of a cEA, comprising selection, recombination mutation and replacement. The only difference to a cEA with static structure is that in this case an individual may find empty nodes in its neighborhood, and the selection pool is restricted to the individuals that occupy adjacent nodes. If at a given time-step an individual has no neighbors, then there is no recombination event for that individual in that specific iteration.

In the position update phase, each individual moves to an adjacent empty node. Adjacency is defined by the Moore neighborhood of radius r , so an individual i at $\rho_g(i) = \langle u, v \rangle$ can move to an empty node $\langle u', v' \rangle$ for which $L_\infty(\langle u, v \rangle, \langle u', v' \rangle) \leq r$. If no empty position is available, the individual stays in the same node. Otherwise, it picks a neighboring empty node according to the marks on them. If there are no marks, the destination is chosen randomly amongst the free nodes.

We consider two possibilities for the position update phase: stimergic, whereby the individual looks for a mark that is similar to itself; and Brownian, whereby the individual selects an empty neighbor regardless of the marks. For the first option, let $\mathcal{N}\langle u, v \rangle = \{\langle u^{(1)}, v^{(1)} \rangle, \dots, \langle u^w, v^w \rangle\}$ be the collection of empty neighboring nodes and let i be the individual to move. Then, the individual attempts to move to a node whose mark is as close as possible to its own corresponding trait (fitness or genotype) or to an adjacent cell picked at random if there are no marks in the neighborhood. This strategy leads to the self-organization of the population into dynamic clusters [3], [8]. In the alternative Brownian policy, the individual moves to an adjacent empty position picked at random. In either case, the process is repeated for the whole population. The following section describes the results attained by dynamic cGAs with stimergic and Brownian movement.

4 Results and Discussion

In order to investigate their performance, the proposed dynamic topologies were tested on a set of functions that challenge the EAs ability to combine building-blocks and demand a careful balance between exploration and exploitation: the near-deceptive order-3 trap, the recursive epistatic H-IFF and the *needle in the haystack* Trident problem.

A trap function is a piecewise-linear function defined on *unitation* (number of ones in a string), with two distinct regions in the search space, one leading to the global optimum and the other leading to a local optimum. The trap in this test is defined by:

$$F(\vec{x}) = \begin{cases} k, & \text{if } u(\vec{x}) = k \\ k - 1 - u(\vec{x}), & \text{otherwise} \end{cases} \quad (1)$$

where $u(\vec{x})$ is the unitation function and k is the problem size (and also the fitness of the global optimum). With these definitions, order-3 traps are in the region between deceptive and non-deceptive, while order-2 are non-deceptive and order-4 are fully deceptive. For the experiments, an order-3 trap function was constructed by juxtaposing 100 subproblems, which corresponds to 300-bit string. The fitness of the best solution (a string of 1's) is 300.

Trident functions are *needle in the haystack* problems that exploit the ability of EAs to mix good but significantly different solutions. The fitness function of the Trident used in this work has two components, *base* and *contribution*: $F(\vec{x}) = \text{base}(\vec{x}) + \text{contribution}(\vec{x})$. The base depends on unitation and is described by:

$$\text{base}(\vec{x}) = \|2 \cdot u(\vec{x}) - l\| \quad (2)$$

where l is the chromosome length and $u(\vec{x})$ is the unitation function. The contribution rewards certain configurations of strings that have an equal number of 0's and 1's. Let L be the first half of the binary string x of length l and R the second half. The *contribution* is described by Equation 5:

$$\text{contribution}(\vec{x}) = \begin{cases} 2 \cdot l, & L = \bar{R} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where \bar{R} is the bitwise negation of R . The Trident accepts strings of length $2k$, where $k \geq 2$. For this paper, 128-bit strings were used and the optimal fitness is 256.

Finally, the H-IFF function is a recursive epistatic problem with hierarchical structure. The landscape requires a search for increasingly higher-order schemata, challenging the EAs' abilities to identify and combine good building blocks. The problem is defined using a recursive function. If the bit string being considered consists of all zeros or all ones, the fitness of the string is equal to its length; otherwise it has a fitness of 0. This same criterion is then applied recursively on each half of the string, until it can be subdivided no further. Adding the fitness of all substrings together yields the fitness of the whole. Formally, the HIFF fitness function can be defined as:

$$f(B) = \begin{cases} 1, & \text{if } |B| = 1 \\ |B| + f(B_L) + f(B_R), & \text{if } |B| > 1 \text{ and } (\forall_i \{b_i = 0\} \text{ or } \forall_i \{b_i = 1\}) \\ f(B_L) + f(B_R), & \text{otherwise} \end{cases} \quad (4)$$

where B is a block of bits ($b_1 \dots b_n$), $|B|$ is the size of the block (and therefore equal to n , which must be an integer power of 2), b_i is the i th element of B , B_L and B_R are

the left and right halves of B. For the tests, a problem 128-bit strings has been constructed. The best solution has a fitness value of 1024.

With this set of functions it is possible to test the ability of the dynamic cGAs in combining the raw building blocks of the initial population and escape local optima traps. These functions challenge standard strategies, which converge very often to local optima, especially in the H-IFF and trap functions. If the proposed dynamic topology is effective in maintaining genetic diversity, then it is expected that the rate of convergence to global optima is improved.

All the cEAs used in the experiments are synchronous (i.e., the offspring are placed in a temporal population and replacement is done after every individual generates one child). Parameterization was done after [1]: the population size was set to $n = 400$; the recombination operator is the double point crossover with $p_c = 1.0$; mutation is bit-flip with $p_m = 1/l$, where l is the chromosome length; tournament selection. Only one offspring is placed in the temporary population (randomly chosen from the set of two children). In the replacement stage, the offspring replaces its parent if it is better.

The stop criteria are: to find the global optimum or to achieve a maximum of 3,000,000 function evaluations. The number of evaluations required to meet the best solution is recorded and averaged over 50 runs. A success measure (successful runs) is defined as the number of runs in which the algorithm attains the global optimum.

Four different strategies have been considered for the position update phase of the proposed algorithm. In the first one, which will be referred to as *dynamic cGA with Brownian movement* ($dcGA_b$) the individuals ignore the marks and chose randomly the destination cells amongst the empty ones in their neighborhood. In the *dynamic cGA with fitness marks* ($dcGA_f$), the individuals deposit marks with their fitness value. A similar strategy is used by the *hierarchical dynamic cGA with fitness marks* ($dcGA_{fh}$), except that in this case the individual only considers a mark if the fitness value is better than its own fitness. Finally, in the *dynamic cGA with genotype marks* ($dcGA_g$), the individuals leave copies of their genotypes in the cells, and when choosing the destination cell, the individual computes the Hamming distance between its genotype and the marks. The destination cell is then the one that minimizes the distance. The radius r of the Moore neighborhood and marks lifespan K were set to 1.

At every time-step, the individuals are ranked according to their fitness, so that the best individuals' positions are updated first. This strategy has been devised for the $DcGA_{fh}$, but in order to make fair comparisons it has been implemented in every cGA. In fact, some preliminary tests showed that ranking the individuals tends to improve the performance of the algorithms.

In order to evaluate the efficiency of the algorithms, the dynamic cGAs are compared with static cGAs with Moore (cGA_M) and von Neumann (cGA_{vN}) neighbourhoods on a 20×20 grid. The evolutionary phase begins only when the average clustering degree k (the number of neighbours of an individual, including the individual itself) rises above 2.5. This *ad hoc* strategy is used for avoiding the initial distribution stage in which many individuals are still isolated (i.e., with none or only a few neighbours). Typically, the individuals start to cluster in a few generations and the evolutionary phase begins at a very early stage. Although the threshold is imposed here by a centralized decision, a local decentralized (self-organized) strategy is also possible. For instance, the evolutionary phase could be triggered individually,

Table 1. Average best fitness values (plus standard deviation)

	X×Y	H-IFF	Trident	3-trap
cGA_M	20×20	877.17±90.95	243.20±38.79	296.58±2.07
cGA_{vN}	20×20	915.84±91.86	243.20±38.79	297.70±1.71
$dcGA_b$	30×30	856.67±100.59	184.32±64.18	295.18±2.30
	40×40	905.33±99.92	235.52±47.40	296.58±1.95
	50×50	902.33±92.51	235.52±47.40	297.78±1.43
	60×60	926.67±102.08	232.96±49.68	297.76±1.70
	70×70	928.96±87.00	219.43±58.42	297.86±1.54
$dcGA_f$	30×30	871.68±109.82	194.56±64.60	294.42±2.82
	40×40	870.08±94.51	230.40±51.72	294.70±3.26
	50×50	917.76±94.67	250.88±25.34	296.54±2.19
	60×60	944.96±90.19	256.00±0.00	297.58±1.48
	70×70	954.88±85.37	256.00±0.00	298.16±1.45
$dcGA_{fh}$	30×30	862.400±97.36	192.00±64.65	294.28±2.38
	40×40	884.80±112.17	235.52±47.40	294.76±2.60
	50×50	921.60±94.80	245.76±35.08	295.90±3.09
	60×60	940.80±93.24	253.44±18.10	296.56±2.55
	70×70	965.12±79.27	248.32±30.71	297.42±1.71
$dcGA_g$	30×30	875.84±92.10	222.72±56.72	295.88±2.16
	40×40	929.60±98.38	253.44±18.10	297.14±1.83
	50×50	924.48±92.86	250.88±25.34	297.86±1.52
	60×60	947.84±84.27	256.00±0.00	298.68±1.12
	70×70	979.84±77.28	256.00±0.00	298.74±1.28

for each chromosome. However, such strategy introduces a transitory phase in which the population only recombines partially (steady-state). This could make a comparison with static strategies more difficult and potentially unfair and therefore it has been left for future work.

The objectives of the first experiment are to study the performance of the dynamic cGAs and the effects of the grid size on their behaviour. For that purpose, grids with different size have been tested, starting with a 30×30 grid. The averaged final fitness value attained by each algorithm in each function is shown in Table 1.

The first conclusion is that the Brownian version, in general, does not improve significantly the performance of the cGA_{vN} (the best static strategy). A dynamic topology *per se* is not sufficient to overcome the drawbacks of standard cEAs. Some kind of organization must take place in order to generate a better interaction between the individuals. When stigmergy is introduced in the model, the results are clearly improved, as seen in Table 1.

The dynamic topologies with stigmergic-based movement rules increase standard cGAs performance when the grid is larger than 40×40 . In general, dynamic populations with stigmergic rule moving on 60×60 and 70×70 grids significantly improve the standard cGAs. For instance, the $dcGA_g$ on a 70×70 grid is significantly better than the standard cGAs in every function (according to Kolmogorov-Smirnov statistical tests with a 0.05 level of significance). The fact that smaller grids do not necessarily improve the static cGA performance suggests that it is not the movement of the individuals that makes the algorithm better in this set of functions, but instead some kind of global island-like pattern that emerges when the grid is larger.

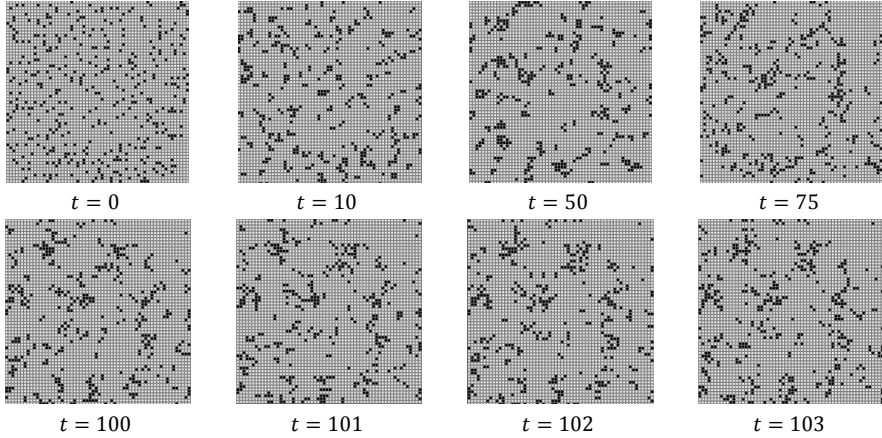


Fig. 1. Distribution of the individuals on the grid at different iterations of the search process. $dcGA_f$ and Trident function. 60×60 grid.

Figure 1 shows the distribution of the individuals at different iterations (t) of the search process for a grid with size 60×60 . The evolutionary phase begins at $t = 50$. Clusters of individuals emerge already at an early stage. Those clusters are highly dynamic and in a few generations the global pattern radically changes (please note the distributions between iteration $t = 100$ and $t = 103$). The topology self-organizes into a kind of dynamic island model, in which the communication between the clusters is also an emergent property, arising from the global behavior of the system. After $t = 50$, when the evolutionary phase is introduced (and therefore several fitness values and genotypes are changing in each time-step), the clusters are sparser, but this is an expected outcome due to the variation introduced by the evolutionary process.

Table 2 shows the number of successful runs attained by the cGAs. Again, under this criterion, the dynamic versions outperform the static topologies. The similarity-based strategy is particularly efficient, attaining the best success rates.

The previous results show that the dynamic cGAs are able to converge more often to global optimum. Therefore, they have a better balance between exploration and exploitation for these fitness functions: with the same raw building-blocks, the dynamic cGAs combine more efficiently the solutions. This is probably because the emergent structures, with their clustering degree and dynamical behaviour, are more efficient at maintaining genetic diversity. In order to investigate this hypothesis,

Table 2. Number of successful runs

	H-IFF	Trident	3-trap
cGA_M	11	45	2
cGA_{vN}	19	45	5
$dcGA_b(70 \times 70)$	21	36	8
$dcGA_f(70 \times 70)$	29	50	9
$dcGA_{fh}(70 \times 70)$	31	48	5
$dcGA_g(70 \times 70)$	37	50	13

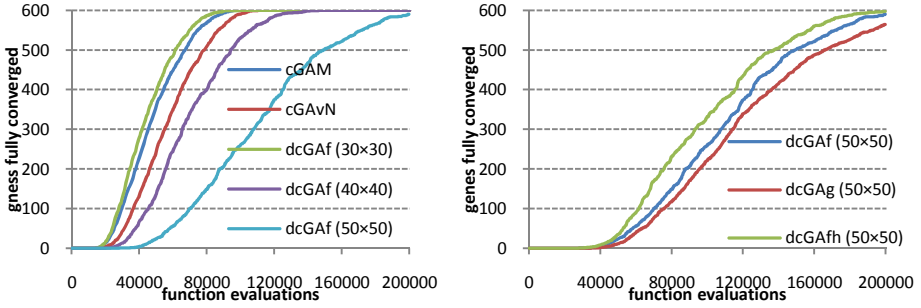


Fig. 2. Genetic diversity

the algorithms were tested without mutation and the number of genes that converge (i.e., genes with alleles 0 or 1 in the entire population) during the run was computed and plotted. The results are in Figure 2. The diversity is in fact maintained at a higher level by the structures. Furthermore, increasing the grid increases the diversity (left-hand graph in Figure 2). As for the different strategies, the best strategy (similarity-based) is also the one that maintains diversity at a higher level (right-hand graph).

Finally, since the dynamic topologies maintain genetic diversity at a higher level, therefore increasing exploration and reducing the risk of convergence to local optima, it is expected that the convergence speed is reduced, a typical payoff for increasing robustness. Table 3 shows the averaged number of evaluations required by each algorithm to reach the global optimum (only runs in which the global optimum has been found are considered). The static structures are faster, but as seen in Table 1 and Table 2, at the expenses of a significant drop of the performance levels.

5 Conclusions and Future Work

This paper investigates a dynamic cellular Genetic Algorithms (cGA) in which the individuals communicate via a grid of nodes and self-organized its structure on that grid. The global behavioral patterns emerge from local interactions defined by simple rules. When compared to static topologies, the dynamic structure maintains genetic diversity at a higher level, resulting in an improvement of the convergence rates to global optimum on a set of functions that defy the GAs abilities to combine building-blocks. Such behavior is attained when the ration between the number of nodes in the grid and the population size is above 4:1. With these settings, the distribution of

Table 3. Convergence speed (function evaluations)

	H-IFF	Trident	3-trap
cGA_M	39600.00±9248.29	44035.56±5678.77	93000.00±3400.00
cGA_{vN}	48191.30±15320.04	47377.78±5265.71	100560.00±13716.21
$dcGA_b$	49808.57±9093.29	55168.47±13663.58	108822.75±4795.37
$dcGA_f$	114608.86±36325.71	75936.81±12991.56	266100.11±48188.63
$dcGA_{fh}$	104568.00±27233.83	69546.29±9204.14	212348.20±20325.13
$dcGA_g$	127981.38±41261.71	95204.48±15756.53	298350.47±20739.19

individuals in the grid emerges into a global island-like model, highly dynamic and with frequent communication between the clusters.

Future work will be focused on the traits of the system and their effects on the behavior of the population and on the performance of the algorithm. Radius r of the neighborhood and marks' lifespan K will be investigated. Different stigmergic strategies will be tested, namely those that favor recombination between dissimilar individuals. Finally, the experiments will be extended to other type of functions (unimodal and multimodal) in order to achieve a better comprehension of the structure's working mechanism and potential as an alternative cGA network.

Acknowledgements. The first author wishes to thank FCT, *Ministério da Ciência e Tecnologia*, his Research Fellowship SFRH/BPD/66876/2009. The work was supported by FCT PROJECT [PEst-OE/EEI/LA0009/2013], Spanish Ministry of Science and Innovation projects TIN2011-28627-C04-02 and TIN2011-28627-C04-01, Andalusian Regional Government P08-TIC-03903 and P10-TIC-6083, CEI-BioTIC UGR project CEI2013-P-14, and UL-EvoPerf project.

References

1. Alba, E., Dorronsoro, B.: The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Trans. Evol. Computation* 9, 126–142 (2005)
2. Cantú-Paz, E.: Migration Policies, Selection Pressure, and Parallel EAs. *Journal of Heuristics*, 311–334 (2001)
3. Fernandes, C.M., Laredo, J.L.L., Merelo, J.J., Cotta, C., Rosa, A.C.: Towards a 2-dimensional Framework for Structured Population-based Metaheuristics. In: *Proc. of IEEE International Conference on Complex Systems*, pp. 1–6 (2012)
4. Fernandes, C.M., Laredo, J.L.L., Merelo, J.J., Cotta, C., Rosa, A.C.: A Study on Time-Varying Partially Connected Topologies for the Particle Swarm. In: *Proc. of the IEEE Congress on Evolutionary Computation*, pp. 2450–2456. IEEE (2013)
5. Grassé, La, P.-P.: reconstruction du nid et les coordinations interindividuelles chez bellicositermes et cubitermes sp. *La théorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs*. *Insectes Sociaux* 6, 41–80 (1959)
6. Ilachinski, A., Cellular Automata, A.: *Discrete Universe*, World Scientific (2001)
7. Laredo, J.L.J., Castillo, P.A., Mora, A.M., Merelo, J.J., Fernandes, C.M.: Resilience to churn of a peer-to-peer evolutionary algorithm. *International Journal of High Performance Systems Architecture* 1(4), 260–268 (2008)
8. Nogueras, R., Cotta, C., Fernandes, C.M., Jiménez Laredo, J.L., Merelo, J.J., Rosa, A.C.: An analysis of a selecto-lamarckian model of multimemetic algorithms with dynamic self-organized topology. In: *Dediu, A.-H., Martín-Vide, C., Truthe, B., Vega-Rodríguez, M.A.* (eds.) *TPNC 2013*. LNCS, vol. 8273, pp. 205–216. Springer, Heidelberg (2013)
9. Steinmetz, R., Wehrle, K. (eds.): *Peer-to-Peer Systems and Applications*. LNCS, vol. 3485. Springer, Heidelberg (2005)
10. Tomassini, M.: *Spatially Structured Evolutionary Algorithms*. Springer (2005)
11. Whitacre, J.M., Sarker, R.A., Pham, Q.: The self-organization of interaction networks for nature-inspired optimization. *IEEE Transactions on Evolutionary Computation* 12, 220–230 (2008)

How to Assess Step-Size Adaptation Mechanisms in Randomised Search

Nikolaus Hansen, Asma Atamna, and Anne Auger

Inria*

LRI (UMR 8623), University of Paris-Sud (UPSud), France

Abstract. Step-size adaptation for randomised search algorithms like evolution strategies is a crucial feature for their performance. The adaptation must, depending on the situation, sustain a large diversity or entertain fast convergence to the desired optimum. The assessment of step-size adaptation mechanisms is therefore non-trivial and often done in too restricted scenarios, possibly only on the sphere function. This paper introduces a (minimal) methodology combined with a practical procedure to conduct a more thorough assessment of the overall population diversity of a randomised search algorithm in different scenarios. We illustrate the methodology on evolution strategies with σ -self-adaptation, cumulative step-size adaptation and two-point adaptation. For the latter, we introduce a variant that abstains from *additional* samples by constructing two particular individuals within the *given* population to decide on the step-size change. We find that results on the sphere function alone can be rather misleading to assess mechanisms to control overall population diversity. The most striking flaws we observe for self-adaptation: on the linear function, the step-size increments are rather small, and on a moderately conditioned ellipsoid function, the adapted step-size is 20 times smaller than optimal.

1 Introduction

In this paper we consider a fitness or objective function, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, to be minimised in a black-box optimisation scenario, and an evolutionary algorithm, or randomised search method, generating λ offspring according to

$$\mathbf{x}_k^{(t)} = \mathbf{x}^{(t)} + \sigma^{(t)} \times \mathbf{z}_k^{(t)}, k = 1, \dots, \lambda, \quad (1)$$

where $\mathbf{x}^{(t)} \in \mathbb{R}^n$ denotes the incumbent solution at iteration t and $\mathbf{z}_k^{(t)} \in \mathbb{R}^n$ are i.i.d. random vectors. The “overall variance” of the offspring population in (1) is determined by the diversity parameter $\sigma^{(t)}$. More generally, we rely on two assumptions: (i) we have a valid measurement for the “global diversity” of the offspring population, denoted as $\sigma^{(t)}$, and (ii) the shape of the offspring population (determined by the distribution of $\mathbf{z}_k^{(t)}$ in (1)) does not change remarkably during the investigated time range of t .

Controlling the overall diversity in the population plays a crucial role in randomised search and has been typically approached by step-size adaptation. Two conflicting objectives are in place. On the one hand, diversity should be as large as possible to prevent

* Research centre Saclay-Île-de-France, TAO team, lastname@lri.fr

premature convergence or convergence to the very next local optimum. On the other hand, fast convergence to a global (or a good local) optimum is desired which is usually accompanied and facilitated by a fast decrease of diversity.

While adaptation of the *shape* of the sample distribution appears to be a solved problem in moderate dimension [6,10,11] (e.g. by CMA), the effective adaptation of the *overall* population diversity seems yet to pose open questions, in particular with recombination or without entire control over the realised distribution. For example, cumulative step-size adaptation is prone to fail when repair or rejection sampling is used.

In this context, we propose a basic *assessment procedure* to evaluate the capability of step-size control, or the entire search algorithm for that matter, to keep the overall diversity, or step-size $\sigma^{(t)}$, within reasonable limits. This procedure might be used during an algorithm designing process, however we like to remind the general scientific principle that a procedure used to systematically *tune parameters* of an algorithm is forfeited to *assess* the resulting algorithm.

In the next section we introduce the assessment methodology. Section 3 introduces the algorithms used in the case study in Section 4. We also introduce a simplified two-point adaptation and tune its damping parameter on the sphere function in Section 3. Section 5 provides a short discussion and summary.

2 Step-Size Evaluation Methodology

General demands on the behaviour of evolutionary algorithms were suggested previously, e.g. in [4,11]. Here, we propose a methodology to specifically investigate and assess the overall population diversity, or step-size, towards meeting reasonable demands via the following scenarios:

Random Fitness (and flat fitness). On the random fitness, all f -values, $f(\mathbf{x})$, are i.i.d., independently of \mathbf{x} as a continuous random variable. For algorithms invariant under order-preserving transformations of f , i.e., algorithms based on f -rankings only (as those investigated in this paper), testing a single continuous f -distribution is sufficient. Generally, we desire stationarity or unbiasedness of parameters under random fitness [11] and here we expect to see an unbiased random walk in log-scale. For the flat fitness, where f is constant, we expect the same behaviour. In contrast, [4] argues for an exponentially increasing step-size on the flat fitness which, however, involves the risk of divergence when the selection pressure is weak [7].

The linear function, where $f : \mathbf{x} \mapsto x_1$ is the prototypical instantiation (see paragraph *Invariance* below). A linear function tests whether and how quickly the diversity can increase. With step-size to zero, any smooth function appears to be an instantiation of the linear function (unless at a local optimum or saddle point) and the diversity should increase in this case. We demand a fast exponential increase, that is, a linear increase on the log-scale [4]. The rate should be at least comparable to the rate of decrease on the sphere function or at least a factor of 1.1 within n evaluations or at least a factor of 2 in n iterations.

The sphere function, $f : \mathbf{x} \mapsto \sum_{i=1}^n x_i^2 = \|\mathbf{x}\|^2$, is the most simple quadratic function, demanding a rapid decrease of the step-size. Arguably, no other function requires a faster step-size decrement. Step-size control should not reduce the fastest possible (optimal) convergence rate on the sphere function by more than a factor of about three.

To achieve linear (i.e. fast) convergence on the sphere function we need to have, at least approximately, $\sigma^{(t)} \propto f(\mathbf{x}^{(t)})^{1/2}$, implying that σ and $f^{1/2}$ converge at the same rate. More specifically, on the sphere function with isotropic sample distribution, there is a constant $\sigma_{\text{opt}}^*(n)$ such that the step-size

$$\sigma^{(t)} = \frac{\sigma_{\text{opt}}^*(n)}{n} \times f(\mathbf{x}^{(t)})^{1/2} \quad (2)$$

achieves optimal convergence speed and $\sigma_{\text{opt}}^*(n) = \Theta(n^0) = \Theta(1)$. When running a real algorithm, the proportionality can only be satisfied in a stochastic sense, i.e. the random variable $\sigma^{(t)}/f(\mathbf{x}^{(t)})^{1/2}$ is stable (for example when $\mathbf{x}^{(t)}/\sigma^{(t)}$ is an irreducible, recurrent and ergodic Markov Chain [3]).

A similar reasoning on $\sigma^{(t)}$ holds true on the ellipsoid function, where the direct link between $\sigma^{(t)}$ and $f(\mathbf{x}^{(t)})^{1/2}$ is less obvious, however presumed in the following to obtain the *optimal* convergence rates to compare with.

The ellipsoid function, $f : \mathbf{x} \mapsto \sum_{i=1}^n \alpha^{(i-1)/(n-1)} x_i^2$, is arguably the most basic function where, for $\alpha \neq 1$, an isotropic distribution of the new offspring is not optimal. The parameter α represents the condition number of the Hessian matrix of f .

With isotropic sample distribution in (1) and $\alpha > 10$, the realised convergence rates are roughly proportional to $10/\alpha$ [12]. Recalling that $f^{1/2}(\mathbf{x}^{(t)})$ and the optimal value for $\sigma^{(t)}$, are linked to each other (Eq. (2)), we observe that with larger α , when approaching the optimum, the optimal step-size changes more slowly (because the realised convergence rate is small). The task to *estimate the optimal step-size* becomes more relevant than the task to *follow the change* of the optimal step-size. In this paper, experiments are done for $\alpha = 1, 10, 100$.

The stationary sphere is an artificial model, resembling the sphere function in that an isotropic sample distribution is optimal, but with *stationary optimal step-size*. While the sphere function tests the ability to decrease the step-size quickly, the stationary sphere function tests the ability to adapt the step-size close to the optimal step-size in the same sphere-like topography without approaching the optimum. With global intermediate or weighted recombination, as used below, the stationary sphere is simulated by setting the norm of the resulting recombination vector (super-parent) to one and re-normalisation of all other individuals or solutions in the algorithm's state by the same factor (see, e.g., lines 5–6 in Algorithm 3). When the population is never reduced to a single point, an appropriate normalisation factor needs to be identified (omitted due to space restrictions). The stationary sphere model is arguably the easiest model for step-size adaptation and we expect to observe close to the optimal step-sizes.

Convergence Rate and Optimal Step-Size. On the last three functions, we compute from a single run with t iterations the consistent estimator

$$\hat{c} = -\frac{1}{T} \sum_{s=t-T}^{t-1} \frac{1}{2} \ln \left(\frac{f(\mathbf{x}^{(s+1)})}{f(\mathbf{x}^{(s)})} \right) \quad (3)$$

for the convergence rate [2, Eq. (24)], where $\mathbf{x}^{(s)} \in \mathbb{R}^n$ is the solution proposed at time step s , and the burn-in time $t-T$ diminishes the possible bias due to initialisation. In this paper we use $T = \lceil t/2 \rceil$, i.e. half of the overall time steps for aggregated measurements. If necessary (e.g., when we terminate due to numerical precision, but want more data), we average \hat{c} over several runs.

We obtain the values for the *optimal* step-size and convergence rate empirically by measuring the convergence rate with $\sigma^{(t)}$ set according to (2) and sweeping through different values for σ_{opt}^* . Generally, we demand the “real” algorithm to perform within a factor of three of this optimal convergence rate, and we prefer larger step-sizes to smaller ones, given the same performance is observed.

Invariance is an important concept in the assessment of algorithms. For example, all linear functions are identical for the below assessed algorithms, because the algorithms are invariant under affine transformations of f and under rotations of the search space. In the case where algorithms do not exhibit certain invariances (e.g. rotation invariance), it is advisable to test different instantiations (e.g. different rotations) of the above scenarios. Scale invariance on the other hand is a prerequisite to measure (3) independently of initial step-size or the distance to the optimum.

We now apply our methodology to three step-size adaptation methods. Due to the space limits, we do not always display single runs, but we consider investigating the evolution of f and σ (both displayed in the log scale) in single runs in all scenarios part of the assessment procedure [15].

3 Considered Step-Size Adaptation Methods

In the following, we consider the $(\mu/\mu, \lambda)$ -ES with weighted recombination [11]. The offspring are generated as in (1) where the i.i.d. $\mathbf{z}_k^{(t)}$ follow the standard multivariate normal distributions, i.e., $\mathbf{z}_k^{(t)} = \mathcal{N}_{t,k}(\mathbf{0}, \mathbf{I})$. They are sorted according to their fitness such that

$$f(\mathbf{x}_{1:\lambda}^{(t)}) \leq f(\mathbf{x}_{2:\lambda}^{(t)}) \leq \dots \leq f(\mathbf{x}_{\lambda:\lambda}^{(t)}) \quad , \quad (4)$$

thereby defining the index $k : \lambda$ used in the following. The μ best individuals are then recombined according to

$$\mathbf{x}^{(t+1)} = \sum_{k=1}^{\mu} w_k \mathbf{x}_{k:\lambda}^{(t)} \quad , \quad (5)$$

where w_k 's are chosen to be optimal on the infinite-dimensional sphere function [1]. We set $\mu = \lfloor \lambda/2 \rfloor$ and therefore have only positive weights while $\lambda = 4 + \lfloor 3 \ln n \rfloor$.

We consider here three ways to adapt the step-size in (1). Self-Adaptation (SA) [14] and Cumulative Step-size Adaptation (CSA) [11] are given in Algorithm 1 and 2. The used default parameter settings for the latter are taken from [9] as $c_\sigma = \frac{\mu_w + 2}{n + \mu_w + 5}$,

Algorithm 1. The $(\mu/\mu, \lambda)$ - σ SA-ES

```

0 given  $n \in \mathbb{N}_+$ ,  $\lambda$ ,  $\mu$ ,  $w_k$ ,  $\tau = 1/\sqrt{2n}$ 
1 initialize  $\mathbf{x}^{(0)} \in \mathbb{R}^n$ ,  $\sigma^{(0)} \in \mathbb{R}_+$ 
2 while not happy
3   if stationary_sphere :
4      $\mathbf{x}^{(t)} = \mathbf{x}^{(t)}/\|\mathbf{x}^{(t)}\|$ 
5   for  $k \in \{1, \dots, \lambda\}$ 
6      $\xi_k^{(t)} = \tau \mathcal{N}_{t,k}(0, \mathbf{I})$ 
7      $\mathbf{z}_k^{(t)} = \mathcal{N}_{t,k}(\mathbf{0}, \mathbf{I})$ 
8      $\sigma_k^{(t)} = \sigma^{(t)} \times \exp(\xi_k^{(t)})$ 
9      $\mathbf{x}_k^{(t)} = \mathbf{x}^{(t)} + \sigma_k^{(t)} \times \mathbf{z}_k^{(t)}$ 
10   $\sigma^{(t+1)} = \sum_{k=1}^{\mu} w_k \sigma_{k:\lambda}^{(t)}$ 
11   $\mathbf{x}^{(t+1)} = \sum_{k=1}^{\mu} w_k \mathbf{x}_{k:\lambda}^{(t)}$ 
12   $t = t + 1$ 

```

Algorithm 2. The $(\mu/\mu, \lambda)$ -CSA-ES

```

0 given  $n \in \mathbb{N}_+$ ,  $\lambda$ ,  $\mu$ ,  $w_k$ ,  $c_\sigma$ ,  $d$ 
1 initialize  $\mathbf{x}^{(0)} \in \mathbb{R}^n$ ,  $\sigma^{(0)} \in \mathbb{R}_+$ ,  $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$ 
2 while not happy
3   if stationary_sphere :
4      $\mathbf{x}^{(t)} = \mathbf{x}^{(t)}/\|\mathbf{x}^{(t)}\|$ 
5   for  $k \in \{1, \dots, \lambda\}$ 
6      $\mathbf{z}_k^{(t)} = \mathcal{N}_{t,k}(\mathbf{0}, \mathbf{I})$ 
7      $\mathbf{x}_k^{(t)} = \mathbf{x}^{(t)} + \sigma^{(t)} \times \mathbf{z}_k^{(t)}$ 
8      $\mathbf{p}_\sigma^{(t+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(t)} +$ 
        $\sqrt{c_\sigma(2 - c_\sigma)/\sum_{k=1}^{\mu} w_k^2} \sum_{k=1}^{\mu} w_k \mathbf{z}_{k:\lambda}^{(t)}$ 
9      $\sigma^{(t+1)} = \sigma^{(t)} \times \exp\left(\frac{c_\sigma}{d} \left(\frac{\|\mathbf{p}_\sigma^{(t+1)}\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right)$ 
10     $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \sigma^{(t)} \sum_{k=1}^{\mu} w_k \mathbf{z}_{k:\lambda}^{(t)}$ 
11     $t = t + 1$ 

```

$d = 1 + 2 \max\left(0, \sqrt{\frac{\mu w - 1}{n+1}} - 1\right) + c_\sigma$. The third method considered for step-size adaptation is presented in the following.

Two-Point Step-Size Adaptation (TPA). We consider a tidied version of Two-Point Step-Size Adaptation (TPA) based on [8,13]. Conceptually, TPA implements a very coarse line search along the direction of the latest mean shift from $\mathbf{x}^{(t-1)}$ to $\mathbf{x}^{(t)}$. In our version, we sample the first two offspring *of the next iteration* along this line. These two offspring are generated as a mirrored pair, symmetric about the current mean vector $\mathbf{x}^{(t)}$,

$$\mathbf{x}_{1/2}^{(t)} = \mathbf{x}^{(t)} \pm \sigma^{(t)} \times \|\mathcal{N}_t(\mathbf{0}, \mathbf{I})\| \frac{\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}}{\|\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}\|}, \quad (6)$$

instead of (1). Their ranking according to the fitness is used to adapt the step-size: if $\mathbf{x}_1^{(t)}$ is better than $\mathbf{x}_2^{(t)}$ the step-size is increased, because there are better points in the direction of the mean shift vector, beyond of where the mean has been moved. Otherwise, the step-size is decreased. By using individuals that are likely to be sampled by the current distribution, information on the “signal strength” is available, because we can compare their fitness to the fitness of the remaining population. Accordingly, we take the difference between the f -ranks of $\mathbf{x}_1^{(t)}$ and $\mathbf{x}_2^{(t)}$ in the population, $\frac{\text{rank}(\mathbf{x}_2^{(t)}) - \text{rank}(\mathbf{x}_1^{(t)})}{\lambda - 1} \in [-1, 1]$. This normalised rank difference is averaged in $s^{(t)}$ and used to finally update the step-size $\sigma^{(t+1)} = \sigma^{(t)} \times \exp(s^{(t)}/d_\sigma)$, where the damping, d_σ , moderates the step-size changes. The details are shown in Algorithm 3.

The constant for which $\sigma^{(t)}$ in (2) achieves optimal convergence rate depends on the sampling. For TPA-like sampling, we denote it $\sigma_{\text{opt TPA}}^*$.

Algorithm 3. The $(\mu/\mu, \lambda)$ -ES with TPA

<pre> 0 given $n \in \mathbb{N}_+$, $\lambda, \mu, c_\sigma = 0.3, d_\sigma = \sqrt{n}, w_k$ 1 init $\mathbf{x}^{(0)} \in \mathbb{R}^n, \sigma^{(0)} \in \mathbb{R}_+, t = 0, s^{(0)} = 0$ 2 while not happy 3 if <i>stationary_sphere</i> : 4 if $t > 0$: 5 $\mathbf{x}^{(t-1)} = \mathbf{x}^{(t-1)} / \ \mathbf{x}^{(t)}\$ 6 $\mathbf{x}^{(t)} = \mathbf{x}^{(t)} / \ \mathbf{x}^{(t)}\$ 7 for $k \in \{1, \dots, \lambda\}$ 8 $\mathbf{z}_k^{(t)} = \mathcal{N}_{t,k}(\mathbf{0}, \mathbf{I})$ 9 if $t > 0$ and $k = 1$: 10 $\mathbf{z}_1^{(t)} = \ \mathcal{N}_t(\mathbf{0}, \mathbf{I})\ \times \frac{(\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)})}{\ \mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}\ }$ </pre>	<pre> 11 if $t > 0$ and $k = 2$: 12 $\mathbf{z}_2^{(t)} = -\mathbf{z}_1^{(t)}$ 13 $\mathbf{x}_k^{(t)} = \mathbf{x}^{(t)} + \sigma^{(t)} \times \mathbf{z}_k^{(t)}$ 14 $\mathbf{x}^{(t+1)} = \sum_{k=1}^{\mu} w_k \mathbf{x}_{k:\lambda}^{(t)}$ 15 if $t > 0$: 16 $s^{(t)} = (1 - c_\sigma) s^{(t-1)} +$ 17 $c_\sigma \frac{\text{rank}(\mathbf{x}_2^{(t)}) - \text{rank}(\mathbf{x}_1^{(t)})}{\lambda - 1}$ 17 $\sigma^{(t+1)} = \sigma^{(t)} \times \exp\left(\frac{s^{(t)}}{d_\sigma}\right)$ 18 $t = t + 1$ </pre>
---	--

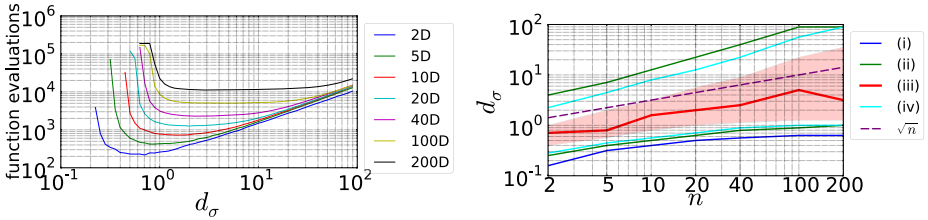


Fig. 1. Left: number of function evaluations versus damping d_σ for TPA, averaged over 101 runs with target f -value 10^{-8} . Right: solid lines depict, from bottom to top, (i) the smallest damping where all runs reached the target value; (ii) the smallest and largest “reasonable” damping with a performance not worse than three times the best (lowest) value in the respective graph on the left; (iii) the damping with best performance, d_σ^* ; (iv) the smallest and largest damping with performance no more than two times worse than the best value in the respective graph on the left, all plotted against dimension. The dashed line depicts \sqrt{n} . The filled area corresponds to damping values with at most 20% performance loss compared to the optimal damping.

The Damping Factor. Here we identify a default value for the damping d_σ . To this aim, we follow a standard procedure: d_σ is tuned on the sphere function. For each value of d_σ , the algorithm is run 101 times with target f -value 10^{-8} (the f -value that stops the algorithm when reached), and if all runs reached the target, the average number of f -evaluations is recorded, see Figure 1, left. We observe a steep incline to the left (small values of d_σ), where missing points indicate the failure of at least one run to reach the target. To the right, the number of f -evaluations increases linearly with the damping and no failures are observed. We extract four damping values per dimension as shown and described in Figure 1, right. We then choose the damping to be (a) more than three times larger than the smallest “reasonable” value and (b) larger than the optimal value such that (c) reducing d_σ by a factor of two leads to a better performance than increasing it by a factor of two without (d) loosing more than a factor of two in performance compared to the best damping (see also [5]). The default choice becomes $d_\sigma = \sqrt{n}$. Note that we

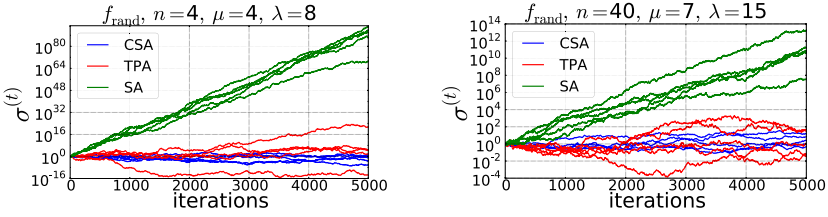


Fig. 2. Evolution of $\sigma^{(t)}$ on the random fitness for 5 runs of SA (green), CSA (blue), and TPA (red) in 4-D (left) and 40-D (right)

identified the damping only for the given default population size. The same procedure needs to be repeated to identify a damping parameter for different population sizes.

4 A Case Study

Experiments are conducted in dimensions between 2 and 100. The algorithms are run with the default parameter settings (Section 3) and initial $\mathbf{x}^{(0)} = (1, 0, \dots, 0)^T$. On random, linear, and ellipsoid function we have $\sigma^{(0)} = 1$, on the sphere and stationary sphere we have $\sigma^{(0)} = \sigma_{\text{opt}}^*/n$ (respectively $\sigma_{\text{opt TPA}}^*/n$) for SA and CSA (respectively TPA). Interquartile ranges are depicted as notched bars with the median at the notch.

Random Fitness. Figure 2 displays the evolution of $\sigma^{(t)}$ for 5000 iterations in 4- and 40-D, five runs for each algorithm. As expected by design, CSA and TPA show an unbiased random walk of $\log \sigma$, where TPA reveals a larger variance. In contrast, due to the combination of geometric mutation and arithmetic recombination of the step-sizes, the random walk of SA is biased [7] and $\log \sigma$ increases linearly with a rate of a little above (below) $10^{0.07} \approx 1.17$ in n iterations for $n = 40$ ($n = 4$, respectively).

Linear Function. On the linear function, the algorithms are run 100 times for 400 iterations. Figure 3 shows geometric average and quartiles of the step-size change realised after n evaluations, $(\sigma^{(t+1)}/\sigma^{(t)})^{n/\lambda}$, compared to results obtained on the random and the sphere function.

For CSA and TPA, the step-size increases by at least a factor of 1.14 within n evaluations. This factor increases slowly with increasing dimension (but never exceeds a factor of two) and the increment on the linear function is at least about three times faster than the decrement on the sphere function.

Self-Adaptation realises only an increment of a factor between 1.03 and 1.05 within n function evaluations, where also decrements appears frequently. The step-size grows faster than on the random function but up to four times slower than it shrinks on the sphere function. This latter observation, together with the observed slow changes rates, fails to meet our original demand.

Sphere. On the sphere function, the target f -value is 10^{-100} . Figure 4 shows nine single runs (left) with $\sigma^{(0)} = 10^{-5}$, the step-size as geometric average (middle), and the convergence rate $\hat{c} \times n/\lambda$ (right, see (3)), both averaged over 100 runs.

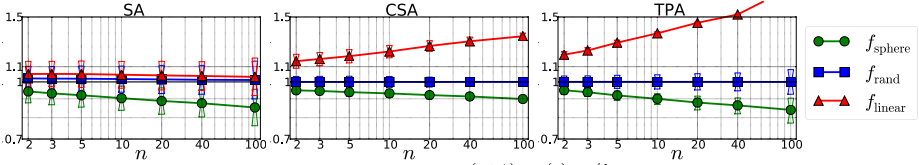


Fig. 3. Step-size change after n evaluations, $(\sigma^{(t+1)}/\sigma^{(t)})^n/\lambda$, on the linear (red), the sphere (green), and the random function (blue)

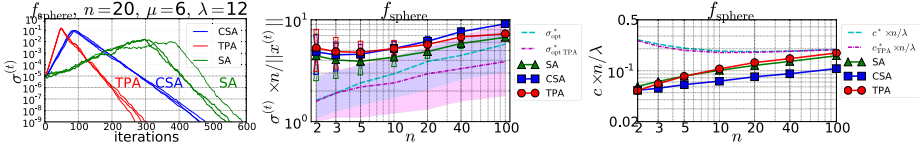


Fig. 4. Single runs (left), step-size (middle) and convergence rate (right) on the sphere function, for SA (green), CSA (blue), and TPA (red) and the respective optimal values. Filled areas correspond to step-size values with at most 20% performance loss compared to σ_{opt} (or σ_{opt} TPA, respectively).

All algorithms realise a too large step-size. In small dimensions, this leads to a loss in performance by about a factor of five, thereby failing our original demand. Fortunately, with increasing dimension the effect diminishes. For $n = 100$, TPA and SA reveal close to optimal convergence rates, whereas CSA is about two times slower.

Supposedly, we observe larger-than-optimal step-sizes, because the optimal step-size changes during the run and is therefore a *moving target*. Indeed, decreasing the damping parameters d or d_σ in CSA or TPA by a factor of two or increasing τ in SA improves the convergence speeds thereby meeting just about the original demand. However for SA, this impairs the performance on the ellipsoid function with $\alpha = 10$.

Ellipsoid. Complementing the observations on the sphere function, which coincides with the ellipsoid function with $\alpha = 1$, the algorithms are investigated on the ellipsoid function with $\alpha \in \{10, 100\}$. These are very moderate condition numbers, where an isotropic distribution can still realise comparatively high convergence rates. We conduct 100 runs with target f -value¹ of 10^{-50} . Figure 5 shows the step-size as geometric average and the convergence rate \hat{c} from (3).

With increasing condition number the realised step-sizes become across the board smaller (compared to the optimal step-size). For $\alpha = 10$, the step-size is still slightly too large with CSA and TPA, while SA shows already too small step-sizes. With $\alpha = 100$, SA realises a 20 times smaller than optimal step-size. Then, for $n \geq 10$, SA performs four to six times slower than optimal, while the other two methods reveal close-to-optimal convergence rates.

Stationary Sphere. On the stationary sphere model, the algorithms are run for $t = 5000$ iterations. The convergence rate \hat{c} from (3) is estimated from 100 runs.

Figure 6 shows step-sizes (as geometric average) and convergence rates. The CSA achieves close to optimal step-sizes and convergence rates in all dimensions. The TPA

¹ In general, we can use such a small target f -value only because the optimum is located at zero and because the distribution shape does not change over the iterations (see Section 1).

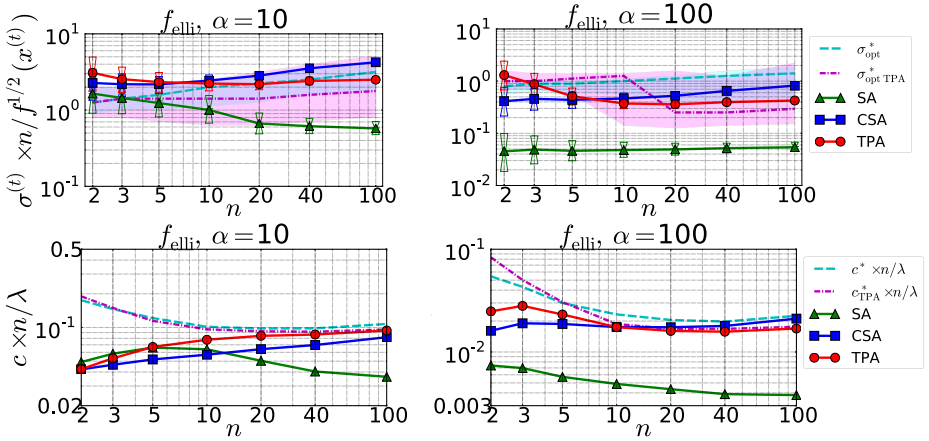


Fig. 5. Results on the ellipsoid with condition number 10 (left) and 100 (right). Top: normalised step-size. Shaded areas depict the step-size range with at most 20% loss in convergence rate. Bottom: convergence rate according to (3).

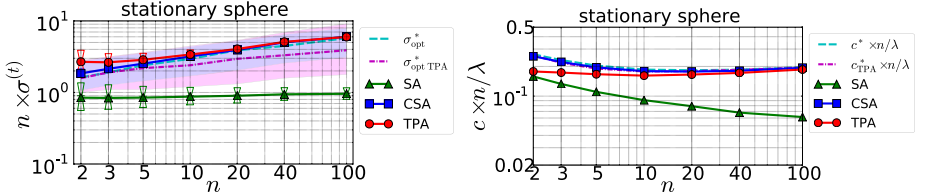


Fig. 6. Step-size (left) and convergence rate (right) of SA (green), CSA (blue), and TPA (red) on the stationary sphere together with the respective optimal values. Shaded areas reflect step-sizes with no more than 20% loss in the achieved convergence rate.

reveals very similar step-sizes in larger dimensions, however for TPA they are somewhat too large, because the optimal step-size is somewhat smaller. Yet, only in smaller dimensions a (moderate) performance loss is observed.

In contrast, SA adapts always a too small step-size. The gap to the optimal step-size is a factor of two in 2-D and increases to a factor of 6 in 100-D. The loss in convergence rate is (slightly) above a factor of three only in 100-D. These observations are (qualitatively) similar to those on the ellipsoid function with condition number 100.

Compared to the sphere function, the observed step-sizes are *in all cases* considerably smaller, again supporting the hypothesis that too large step-sizes are observed on the sphere function mainly because the optimal step-size is a moving target².

5 Discussion and Summary

We have introduced a methodology to assess the overall population diversity, for example determined via step-size adaptation, by describing the desired outcomes on basic

² Experiments with varying damping- or τ -values give additional strong support. Increasing damping impairs the performance on the sphere function (cp. Fig. 1) by reducing the change rate of the step-size, while it (slightly) improves the performance on the stationary sphere.

scenarios. We conducted a case study assessing evolution strategies with weighted recombination and three different step-size adaptation mechanisms.

Despite the small number of investigated algorithms, we find in each test scenario, arguably with exception of the random function, limitations of at least one method: a (too) slow step-size increase on the linear function; a (too) slow step-size decrease on the sphere function in small dimensions; adaptation of a far too small step-size on the ellipsoid and stationary sphere. The results suggest that both, design and assessment of step-size adaptation methods is more intricate than one would have hoped for.

Acknowledgments. This work was supported by the grant ANR-2012-MONU-0009 (NumBBO) of the French National Research Agency.

References

1. Arnold, D.V.: Optimal weighted recombination. In: Wright, A.H., Vose, M.D., De Jong, K.A., Schmitt, L.M. (eds.) FOGA 2005. LNCS, vol. 3469, pp. 215–237. Springer, Heidelberg (2005)
2. Auger, A., Hansen, N.: Reconsidering the progress rate theory for evolution strategies in finite dimensions. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 445–452. ACM (2006)
3. Auger, A., Hansen, N.: On Proving Linear Convergence of Comparison-based Step-size Adaptive Randomized Search on Scaling-Invariant Functions via Stability of Markov Chains (2013) ArXiv eprint
4. Beyer, H.-G., Deb, K.: On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 5(3), 250–270 (2001)
5. Brockhoff, D., Auger, A., Hansen, N., Arnold, D.V., Hohm, T.: Mirrored sampling and sequential selection for evolution strategies. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 11–21. Springer, Heidelberg (2010)
6. Glasmachers, T., Schaul, T., Yi, S., Wierstra, D., Schmidhuber, J.: Exponential natural evolution strategies. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 393–400. ACM (2010)
7. Hansen, N.: An analysis of mutative σ -self-adaptation on linear fitness functions. *Evolutionary Computation* 14(3), 255–275 (2006)
8. Hansen, N.: CMA-ES with two-point step-size adaptation. CoRR, abs/0805.0231 (2008)
9. Hansen, N.: The CMA evolution strategy: A tutorial (2011)
10. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiño, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004)
11. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
12. Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A.: Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Applied Soft Computing* 11(8), 5755–5769 (2011)
13. Salomon, R.: Evolutionary algorithms and gradient search: Similarities and differences. *IEEE Transactions on Evolutionary Computation* 2(2), 45–55 (1998)
14. Schwefel, H.-P.: *Evolution and Optimum Seeking*. In: *Sixth-Generation Computer Technology*, Wiley Interscience, New York (1995)
15. <http://hal.inria.fr/hal-00997294>

Maximum Likelihood-Based Online Adaptation of Hyper-Parameters in CMA-ES

Ilya Loshchilov¹, Marc Schoenauer^{2,3}, Michèle Sebag^{3,2},
and Nikolaus Hansen^{2,3}

¹ Laboratory of Intelligent Systems,
École Polytechnique Fédérale de Lausanne, Switzerland
`Ilya.Loshchilov@epfl.ch`

² TAO Project-team, INRIA Saclay - Île-de-France

³ Laboratoire de Recherche en Informatique (UMR CNRS 8623)
Université Paris-Sud, 91128 Orsay Cedex, France
`FirstName.LastName@inria.fr`

Abstract. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is widely accepted as a robust derivative-free continuous optimization algorithm for non-linear and non-convex optimization problems. CMA-ES is well known to be almost parameterless, meaning that only one hyper-parameter, the population size, is proposed to be tuned by the user. In this paper, we propose a principled approach called self-CMA-ES to achieve the online adaptation of CMA-ES hyper-parameters in order to improve its overall performance. Experimental results show that for larger-than-default population size, the default settings of hyper-parameters of CMA-ES are far from being optimal, and that self-CMA-ES allows for dynamically approaching optimal settings.

1 Introduction

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES [5]) is a continuous optimizer which only exploits the ranking of estimated candidate solutions to approach the optimum of an objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. CMA-ES is also invariant w.r.t. affine transformations of the decision space, explaining the known robustness of the algorithm. An important practical advantage of CMA-ES is that all hyper-parameters thereof are defined by default with respect to the problem dimension n . Practically, only the population size λ is suggested to be tuned by the user, e.g. when a parallelization of the algorithm is considered or the problem at hand is known to be multi-modal and/or noisy [1,8]. Other hyper-parameters have been provided robust default settings (depending on n and λ), in the sense that their offline tuning allegedly hardly improves the CMA-ES performance for unimodal functions. In the meanwhile, for multi-modal functions it is suggested that the overall performance can be significantly improved by offline tuning of λ and multiple stopping criteria [16,11]. Additionally, it is shown that CMA-ES can be improved by a factor up to 5-10 by the use of surrogate models on unimodal ill-conditioned functions [14]. This suggests that the CMA-ES performance can be improved by better exploiting the information in the evaluated samples $(\mathbf{x}, f(\mathbf{x}))$.

This paper focuses on the automatic online adjustment of the CMA-ES hyper-parameters. The proposed approach, called self-CMA-ES, relies on a second CMA-ES instance operating on the hyper-parameter space of the first CMA-ES, and aimed at increasing the likelihood of generating the most successful samples \mathbf{x} in the current generation. The paper is organized as follows. Section 2 describes the original $(\mu/\mu_w, \lambda)$ -CMA-ES. self-CMA-ES is described in Section 3 and its experimental validation is discussed in Section 4 comparatively to related work. Section 5 concludes the paper.

2 Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy [6,7,5] is acknowledgedly the most popular and the most efficient Evolution Strategy algorithm.

The original $(\mu/\mu_w, \lambda)$ -CMA-ES (Algorithm 1) proceeds as follows. At the t -th iteration, a Gaussian distribution $\mathcal{N}(\mathbf{m}^t, \sigma^{t^2} \mathbf{C}^t)$ is used to generate λ candidate solution $\mathbf{x}_k \in \mathbb{R}^n$, for $k = 1 \dots \lambda$ (line 5):

$$\mathbf{x}_k^t = \mathcal{N}(\mathbf{m}^t, \sigma^{t^2} \mathbf{C}^t) = \mathbf{m}^t + \sigma^t \mathcal{N}(\mathbf{0}, \mathbf{C}^t), \quad (1)$$

where the mean $\mathbf{m}^t \in \mathbb{R}^n$ of the distribution can be interpreted as the current estimate of the optimum of function f , $\mathbf{C}^t \in \mathbb{R}^{n \times n}$ is a (positive definite) covariance matrix and σ^t is a mutation step-size. These λ solutions are evaluated according to f (line 6). The new mean \mathbf{m}^{t+1} of the distribution is computed as a *weighted sum* of the best μ individuals out of the λ ones (line 7). Weights $w_1 \dots w_\mu$ are used to control the impact of selected individuals, with usually higher weights for top ranked individuals (line 1).

The adaptation of the step-size σ^t , inherited from the Cumulative Step-Size Adaptation Evolution Strategy (CSA-ES [6]), is controlled by the evolution path \mathbf{p}_σ^{t+1} . Successful mutation steps $\frac{\mathbf{m}^{t+1} - \mathbf{m}^t}{\sigma^t}$ (line 8) are tracked in the sampling space, i.e., in the isotropic coordinate system defined by the eigenvectors of the covariance matrix \mathbf{C}^t . To update the evolution path \mathbf{p}_σ^{t+1} , i) a decay/relaxation factor c_σ is used to decrease the importance of previous steps; ii) the step-size is increased if the length of the evolution path \mathbf{p}_σ^{t+1} is longer than the expected length of the evolution path under random selection $\mathbb{E} \|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$; iii) otherwise it is decreased (line 13). Expectation of $\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$ is approximated by $\sqrt{n}(1 - \frac{1}{4n} + \frac{1}{21n^2})$. A damping parameter d_σ controls the change of the step-size.

The covariance matrix update consists of two parts (line 12): a *rank-one update* [7] and a *rank- μ update* [5]. The rank-one update computes the evolution path \mathbf{p}_c^{t+1} of successful moves of the mean $\frac{\mathbf{m}^{t+1} - \mathbf{m}^t}{\sigma^t}$ of the mutation distribution in the given coordinate system (line 10), along the same lines as the evolution path \mathbf{p}_σ^{t+1} of the step-size. To stall the update of \mathbf{p}_c^{t+1} when σ increases rapidly, a h_σ trigger is used (line 9).

The rank- μ update computes a covariance matrix \mathbf{C}_μ as a weighted sum of the covariances of successful steps of the best μ individuals (line 11). Covariance matrix

Algorithm 1. The $(\mu/\mu_w, \lambda)$ -CMA-ES

1. **given** $n \in \mathbb{N}_+$, $\lambda = 4 + \lfloor 3 \ln n \rfloor$, $\mu = \lfloor \lambda/2 \rfloor$, $w_i = \frac{\ln(\mu + \frac{1}{2}) - \ln i}{\sum_{j=1}^{\mu} (\ln(\mu + \frac{1}{2}) - \ln j)}$ for $i = 1 \dots \mu$,
 $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$, $c_\sigma = \frac{\mu_w + 2}{n + \mu_w + 3}$, $d_\sigma = 1 + c_\sigma + 2 \max(0, \sqrt{\frac{\mu_w - 1}{n+1}} - 1)$, $c_c = \frac{4}{n+4}$,
 $c_1 = \frac{2}{(n+1.3)^2 + \mu_w}$, $c_\mu = \frac{2(\mu_w - 2 + 1/\mu_w)}{(n+2)^2 + \mu_w}$
 2. **initialize** $\mathbf{m}^{t=0} \in \mathbb{R}^n$, $\sigma^{t=0} > 0$, $\mathbf{p}_\sigma^{t=0} = \mathbf{0}$, $\mathbf{p}_c^{t=0} = \mathbf{0}$, $\mathbf{C}^{t=0} = \mathbf{I}$, $t \leftarrow 0$
 3. **repeat**
 4. **for** $k = 1, \dots, \lambda$ **do**
 5. $\mathbf{x}_k = \mathbf{m}^t + \sigma^t \mathcal{N}(\mathbf{0}, \mathbf{C}^t)$
 6. $\mathbf{f}_k = f(\mathbf{x}_k)$
 7. $\mathbf{m}^{t+1} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$ // the symbol $i : \lambda$ denotes i -th best individual on f
 8. $\mathbf{p}_\sigma^{t+1} = (1 - c_\sigma) \mathbf{p}_\sigma^t + \sqrt{c_\sigma(2 - c_\sigma)} \sqrt{\mu_w} \mathbf{C}^{t-\frac{1}{2}} \frac{\mathbf{m}^{t+1} - \mathbf{m}^t}{\sigma^t}$
 9. $h_\sigma = \mathbb{1} \left\| \mathbf{p}_\sigma^{t+1} \right\| < \sqrt{1 - (1 - c_\sigma)^{2(t+1)}} (1.4 + \frac{2}{n+1}) \mathbb{E} \|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$
 10. $\mathbf{p}_c^{t+1} = (1 - c_c) \mathbf{p}_c^t + h_\sigma \sqrt{c_c(2 - c_c)} \sqrt{\mu_w} \frac{\mathbf{m}^{t+1} - \mathbf{m}^t}{\sigma^t}$
 11. $\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i \frac{\mathbf{x}_{i:\lambda} - \mathbf{m}^t}{\sigma^t} \times \frac{(\mathbf{x}_{i:\lambda} - \mathbf{m}^t)^T}{\sigma^t}$
 12. $\mathbf{C}^{t+1} = (1 - c_1 - c_\mu) \mathbf{C}^t + c_1 \underbrace{\mathbf{p}_c^{t+1} \mathbf{p}_c^{t+1T}}_{\text{rank-one update}} + c_\mu \underbrace{\mathbf{C}_\mu}_{\text{rank-}\mu \text{ update}}$
 13. $\sigma^{t+1} = \sigma^t \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\mathbf{p}_\sigma^{t+1}\|}{\mathbb{E} \|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right)$
 14. $t = t + 1$
 15. **until** *stopping criterion is met*
-

\mathbf{C} itself is replaced by a weighted sum of the rank-one (weight c_1 [7]) and rank- μ (weight c_μ [5]) updates, with c_1 and c_μ positive and $c_1 + c_\mu \leq 1$.

While the optimal parameterization of CMA-ES remains an open problem, the default parameterization is found quite robust on noiseless unimodal functions [5], which explains the popularity of CMA-ES.

3 The Self-CMA-ES

The proposed self-CMA-ES approach is based on the intuition that the optimal hyper-parameters of CMA-ES at time t should favor the generation of the best individuals at time t , under the (strong) assumption that an optimal parameterization and performance of CMA-ES in each time t will lead to the overall optimal performance.

Formally, this intuition leads to the following procedure. Let θ_f^t denote the hyper-parameter vector used for the optimization of objective f at time t (CMA-ES stores its state variables and internal parameters of iteration t in θ^t and the \cdot^t -notation is used to access them). At time $t + 1$, the best individuals generated according to θ_f^t are known to be the top-ranked individuals $\mathbf{x}_{1:\lambda}^t \dots \mathbf{x}_{\mu:\lambda}^t$, where $\mathbf{x}_{i:\lambda}^t$ stands for the i -th best individual w.r.t. f . Hyper-parameter vector θ_f^t would thus have been all the better, if it had maximized the probability of generating these top individuals.

Along this line, the optimization of θ_f^t is conducted using a second CMA-ES algorithm, referred to as *auxiliary* CMA-ES as opposed to the one concerned with the optimization of f , referred to as *primary* CMA-ES. The objective of the auxiliary CMA-ES is specified as follows:

Given: hyper-parameter vector θ_f^i and points $(\mathbf{x}_{1:\lambda}^i, f(\mathbf{x}_{1:\lambda}^i))$ evaluated by primary CMA-ES at steps $i = 1, \dots, t$ (noted as $\theta_f^{i+1} \cdot f(\mathbf{x}_{1:\lambda})$ in Algorithm 2),

Find: $\theta_f^{t,*}$ such that i) backtracking the primary CMA-ES to its state at time $t-1$; ii) replacing θ_f^t by $\theta_f^{t,*}$, would maximize the likelihood of $\mathbf{x}_{i:\lambda}^t$ for $i = 1 \dots \mu$.

The auxiliary CMA-ES might thus tackle the maximization of $g_t(\theta)$ computed as the weighed log-likelihood of the top-ranked μ_{sel} individuals at time t :

$$g_t(\theta) = \sum_{i=1}^{\mu_{sel}} w_{sel,i} \log(P(\mathbf{x}_{i:\lambda}^t | \theta_f^t = \theta)), \quad (2)$$

where $w_{sel,i} \geq 0$, $i = 1 \dots \mu_{sel}$, $\sum_{i=1}^{\mu_{sel}} w_{sel,i} = 1$, and by construction

$$P(\mathbf{x}_i | \mathbf{m}^t, \mathbf{C}^t) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{C}^t|}} \exp(-0.5(\mathbf{m}^t - \mathbf{x}_i^t) \mathbf{C}^{t-1} (\mathbf{m}^t - \mathbf{x}_i^t)), \quad (3)$$

where \mathbf{C}^t is the covariance matrix multiplied by σ^{t2} and $|\mathbf{C}^t|$ is its determinant.

While the objective function for the auxiliary CMA-ES defined by Eq. (2) is mathematically sound, it yields a difficult optimization problem; firstly the probabilities are scale-sensitive; secondly and overall, in a worst case scenario, a single good but unlikely solution may lead the optimization of θ_f^t astray.

Therefore, another optimization objective $h_t(\theta)$ is defined for the auxiliary CMA-ES, where $h_t(\theta)$ measures the agreement on $\mathbf{x}_{i:\lambda}^t$ for $i = 1 \dots \mu$ between i) the order defined from f ; ii) the order defined from their likelihood conditioned by $\theta_f^t = \theta$ (Algorithm 3). Procedure *ReproduceGenerationCMA* in Algorithm 3 updates the strategy parameters described from line 7 to line 14 in Algorithm 1 using already evaluated solutions stored in $\theta_f^t \cdot \mathbf{x}_{i:\lambda}$. Line 4 computes the Mahalanobis distance, division by step-size is not needed since only ranking will be considered in line 5 (decreasing order of Mahalanobis distances corresponds to increasing order of log-likelihoods). Line 6 computes a weighted sum of ranks of likelihoods of best individuals.

Finally, the overall scheme of self-CMA-ES (Algorithm 2) involves two interdependent CMA-ES optimization algorithms, where the primary CMA-ES is concerned with optimizing objective f , and the auxiliary CMA-ES is concerned with optimizing objective h_t , that is, optimizing the hyper-parameters of the primary CMA-ES¹. Note that self-CMA-ES is not *per se* a “more parameterless” algorithm than CMA-ES, in the sense that the user is still invited to modify the population size λ . The main purpose of self-CMA-ES is to achieve the online adaptation of the other CMA-ES hyper-parameters.

¹ This scheme is actually inspired from the one proposed for surrogate-assisted optimization [13], where the auxiliary CMA-ES was in charge of optimizing the surrogate learning hyper-parameters.

Algorithm 2. The self-CMA-ES

-
1. $t \leftarrow 1$
 2. $\theta_f^t \leftarrow \text{InitializationCMA}()$ { primary CMA-ES aimed at optimizing f }
 3. $\theta_h^t \leftarrow \text{InitializationCMA}()$ { auxiliary CMA-ES aimed at optimizing h_t }
 4. fill θ_f^t with corresponding parameters stored in mean of distribution $\theta_h^t.m$
 5. $\theta_f^{t+1} \leftarrow \text{GenerationCMA}(f, \theta_f^t)$
 6. $t \leftarrow t + 1$
 7. **repeat**
 8. $\theta_f^{t+1} \leftarrow \text{GenerationCMA}(f, \theta_f^t)$
 9. $\theta_h^{t+1} \leftarrow \text{GenerationCMA}(h_t, \theta_h^t)$
 10. fill θ_f^{t+1} with corresponding parameters stored in mean of distribution $\theta_h^{t+1}.m$
 11. $t \leftarrow t + 1$
 12. **until** stopping criterion is met
-

Specifically, while the primary CMA-ES optimizes $f(\mathbf{x})$ (line 8), the auxiliary CMA-ES maximizes $h_t(\theta)$ (line 9) by sampling and evaluating λ_h variants of θ_f^t . The updated mean of the auxiliary CMA-ES in the hyper-parameter space is used as a local estimate of the optimal hyper-parameter vector for the primary CMA-ES. Note that the auxiliary CMA-ES achieves a single iteration in the hyper-parameter space of the primary CMA-ES, with two motivations: limiting the computational cost of self-CMA-ES (which scales as λ_h times the time complexity of the CMA-ES), and preventing θ_f^t from overfitting the current sample $\mathbf{x}_{i:\lambda}^t, i = 1 \dots \mu$.

Algorithm 3. Objective function $h_t(\theta)$

-
1. **Input:** $\theta, \theta_f^{t+1}, \theta_f^{t-1}, \theta_f^t, \mu, w_{sel,i}$ for $i = 1, \dots, \mu$
 2. $\theta'^{t-1} \leftarrow \theta$
 3. $\theta_f^t \leftarrow \text{ReproduceGenerationCMA}(f, \theta_f^{t-1})$ using already evaluated $\theta_f^t.\mathbf{x}_{i:\lambda}$
 4. $d_i \leftarrow \left\| \theta_f^t \cdot \sqrt{C^{-1}} \cdot (\theta_f^{t+1}.\mathbf{x}_i^t - \theta_f^t.m) \right\|$; for $i = 1, \dots, \theta_f^{t+1}.\lambda$
 5. $p_i \leftarrow \text{rank of } d_i, i = 1 \dots \lambda$ sorted in decreasing order
 6. $h(\theta) \leftarrow \sum_{i=1}^{\mu} w_{sel,i} p_{i:\lambda}$ { $i : \lambda$ denotes the rank of $\theta^{t+1}.\mathbf{x}_i$ }
 7. **Output:** $h(\theta)$
-

4 Experimental Validation

The experimental validation of self-CMA-ES investigates the performance of the algorithm comparatively to CMA-ES on the BBOB noiseless problems [4]. Both algorithms are launched in IPOP scenario of restarts when the CMA-ES is restarted with doubled population size once stopping criteria [3] are met².

² For the sake of reproducibility, the source code is available at

<https://sites.google.com/site/selfcmappsn/>

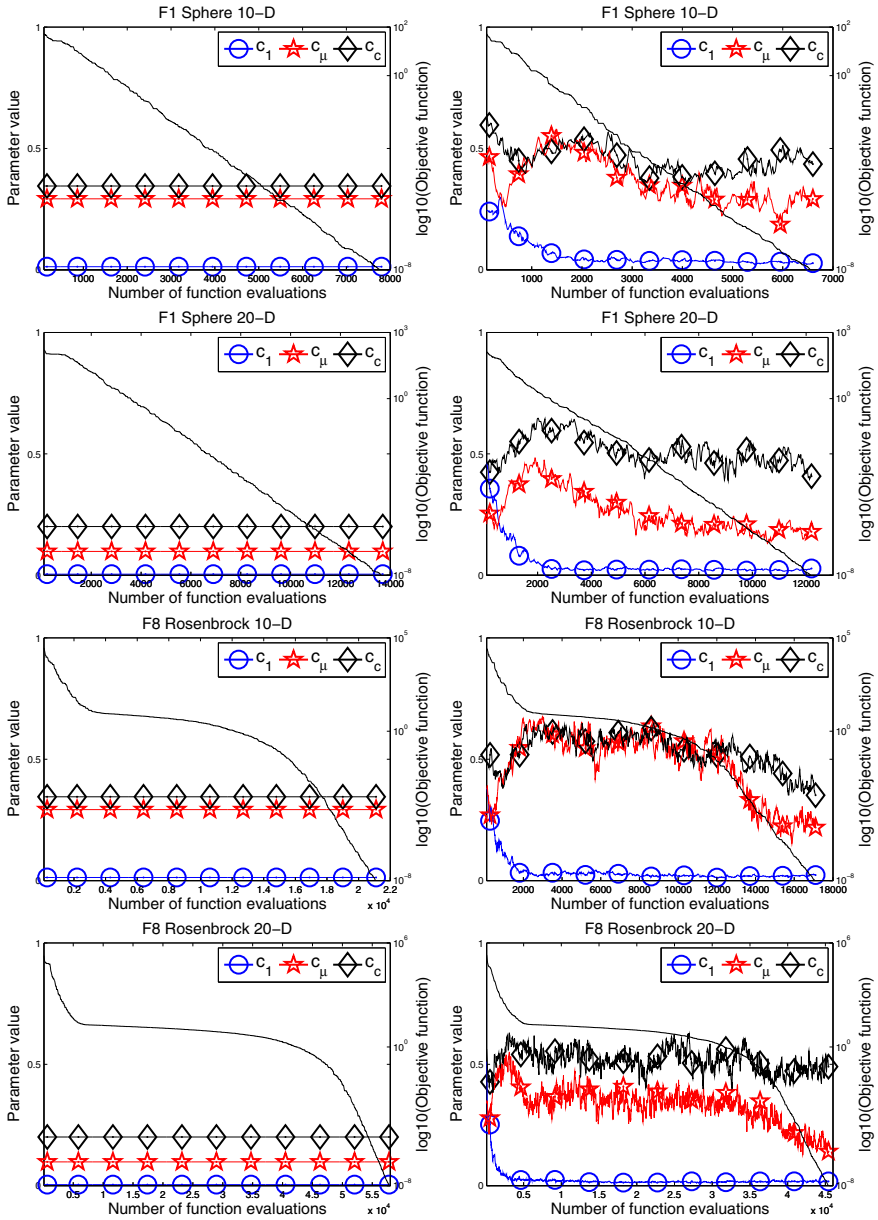


Fig. 1. Evolution of learning rates c_1 , c_μ , c_c (lines with markers, left y-axis) and \log_{10} (objective function) (plain line, right y-axis) of CMA-ES (left column) and self-CMA-ES (right column) on 10- and 20-dimensional Sphere and Rosenbrock functions from [4]. The medians of 15 runs are shown.

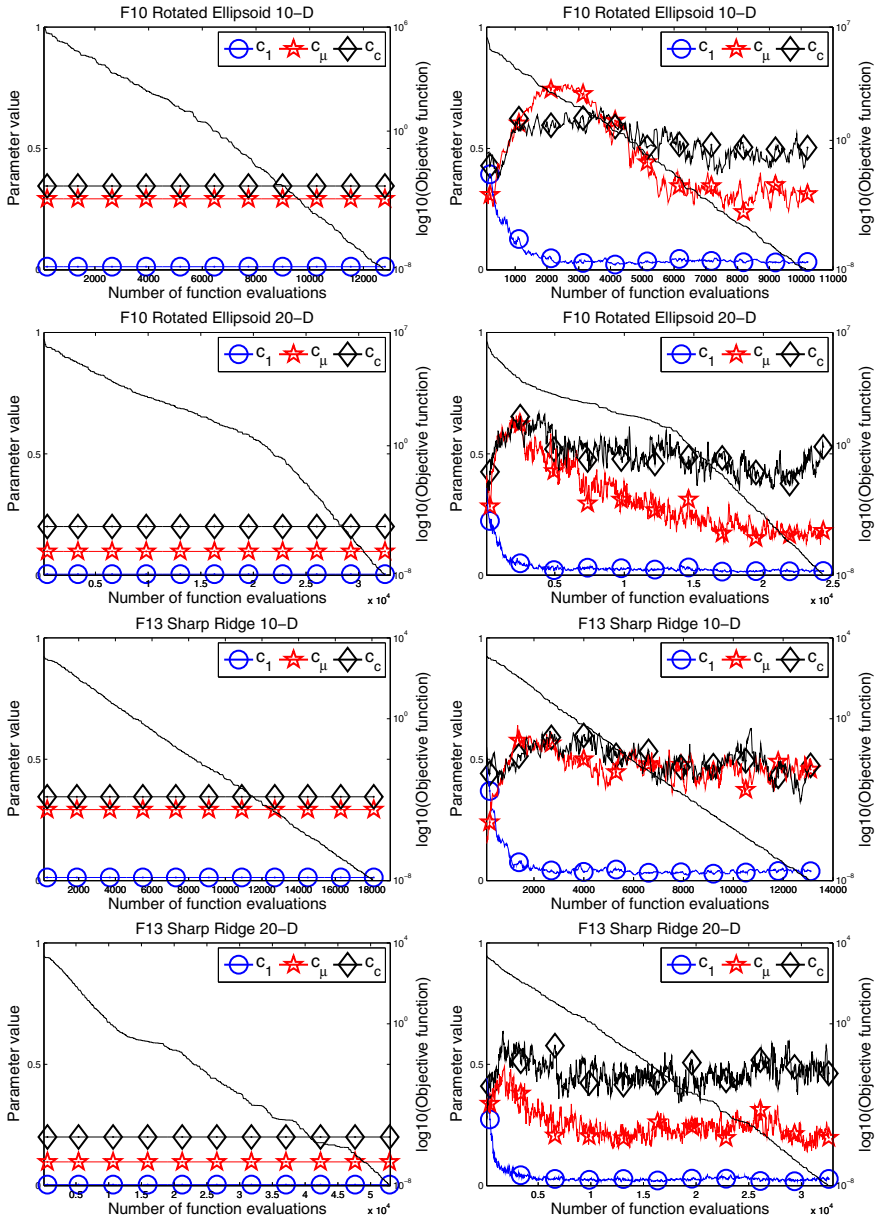


Fig. 2. Evolution of learning rates c_1 , c_μ , c_c (lines with markers, left y-axis) and $\log_{10}(\text{objective function})$ (plain line, right y-axis) of CMA-ES (left column) and self-CMA-ES (right column) on 10- and 20-dimensional Rotated Ellipsoid and Sharp Ridge functions from [4]. The medians of 15 runs are shown.

The population size λ is chosen to be 100 for both CMA-ES and self-CMA-ES. We choose this value (about 10 times larger than the default one, see the default parameters of CMA-ES in Algorithm 1) to investigate how sub-optimal the other CMA-ES hyper-parameters, derived from λ , are in such a case, and whether self-CMA-ES can recover from this sub-optimality.

The auxiliary CMA-ES is concerned with optimizing hyper-parameters c_1 , c_μ and c_c (Algorithm 1), responsible for the adaptation of the covariance matrix of the primary CMA-ES. These parameters range in $[0, .9]$ subject to $0 \leq c_1 + c_\mu \leq 0.9$; the constraint is meant to enforce a feasible \mathbf{C} update for the primary CMA-ES (the decay factor of \mathbf{C} should be in $[0, 1]$). Infeasible hyper-parameter solutions get a very large penalty, multiplied by the sum of distances of infeasible hyper-parameters to the range of feasibility.

We set $w_{sel,i} = 1/\mu$ for $i = 1, \dots, \mu$ and $\mu = \lfloor \lambda/2 \rfloor$ to 50. The internal computational complexity of self-CMA-ES thus is $\lambda_h = 20$ times larger than the one of CMA-ES without lazy update (being reminded that the internal time complexity is usually negligible compared to the cost per objective function evaluation).

4.1 Results

Figures 1 and 2 display the comparative performances of CMA-ES (left) and self-CMA-ES (right) on 10 and 20-dimensional Sphere, Rosenbrock, Rotated Ellipsoid and Sharp ridge functions from the noiseless BOB testbed [4] (medians out of 15 runs). Each plot shows the value of the hyper-parameters (left y-axis) together with the objective function (in logarithmic scale, right y-axis). Hyper-parameters c_1 , c_μ and c_c are constant and set to their default values for CMA-ES while they are adapted along evolution for self-CMA-ES.

In self-CMA-ES, the hyper-parameters are uniformly initialized in $[0, 0.9]$ (therefore the medians are close to 0.45) and they gradually converge to values which are estimated to provide the best update of the covariance matrix w.r.t. the ability to generate the current best individuals. It is seen that these values are problem and dimension-dependent. The values of c_1 are always much smaller than c_μ but are comparable to the default c_1 . The values of c_μ and c_c and c_1 are almost always larger than the default ones; this is not a surprise for c_1 and c_μ , as their original default values are chosen in a rather conservative way to prevent degeneration of the covariance matrix.

Several interesting observations can be made about the dynamics of the parameter values. The value of c_μ is high most of the times on the Rosenbrock functions, but it decreases toward values similar to those of the Sphere functions, when close to the optimum. This effect is observed on most problems; indeed, on most problems fast adaptation of the covariance matrix will improve the performance in the beginning, while the distribution shape should remain stable when the covariance matrix is learned close to the optimum.

The overall performance of self-CMA-ES on the considered problems is comparable to that of CMA-ES, with a speed-up of a factor up to 1.5 on Sharp Ridge functions. The main result is the ability of self-CMA-ES to achieve the

online adaptation of the hyper-parameters depending on the problem at hand, side-stepping the use of long calibrated default settings³.

4.2 Discussion

self-CMA-ES offers a proof of concept for the online adaptation of three CMA-ES hyper-parameters in terms of feasibility and usefulness. Previous studies on parameter settings for CMA-ES mostly considered offline tuning (see, e.g., [16,11]) and theoretical analysis dated back to the first papers on Evolution Strategies. The main limitation of these studies is that the suggested hyper-parameter values are usually specific to the (class of) analyzed problems. Furthermore, the suggested values are fixed, assuming that optimal parameter values remain constant along evolution. However, when optimizing a function whose landscape gradually changes when approaching the optimum, one may expect optimal hyper-parameter values to reflect this change as well.

Studies on the online adaptation of hyper-parameters (apart from σ , \mathbf{m} and \mathbf{C}) usually consider population size in noisy [2], multi-modal [1,12] or expensive [9] optimization. A more closely related approach was proposed in [15] where the learning rate for step-size adaptation is adapted in a stochastic way similarly to Rprop-updates [10].

5 Conclusion and Perspectives

This paper proposes a principled approach for the self-adaptation of CMA-ES hyper-parameters, tackled as an auxiliary optimization problem: maximizing the likelihood of generating the best sampled solutions. The experimental validation of self-CMA-ES shows that the learning rates involved in the covariance matrix adaptation can be efficiently adapted on-line, with comparable or better results than CMA-ES. It is worth emphasizing that matching the performance of CMA-ES, the default setting of which represent a historical consensus between theoretical analysis and offline tuning, is nothing easy.

The main novelty of the paper is to offer an intrinsic assessment of the algorithm internal state, based on retrospective reasoning (given the best current solutions, how could the generation of these solutions have been made easier) and on one assumption (the optimal hyper-parameter values at time t are "sufficiently good" at time $t + 1$). Further work will investigate how this intrinsic assessment can support the self-adaptation of other continuous and discrete hyper-parameters used to deal with noisy, multi-modal and constrained optimization problems.

Acknowledgments. We acknowledge anonymous reviewers for their constructive comments. This work was supported by the grant ANR-2010-COSI-002 (SIMINOLE) of the French National Research Agency.

³ $c_c = \frac{4}{n+4}$, $c_1 = \frac{2}{(n+1.3)^2 + \mu_w}$, $c_\mu = \frac{2(\mu_w - 2 + 1/\mu_w)}{(n+2)^2 + \mu_w}$.

References

1. Auger, A., Hansen, N.: A Restart CMA Evolution Strategy With Increasing Population Size. In: IEEE Congress on Evolutionary Computation, pp. 1769–1776. IEEE Press (2005)
2. Beyer, H.-G., Hellwig, M.: Controlling population size and mutation strength by meta-es under fitness noise. In: Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII, FOGA XII 2013, pp. 11–24. ACM (2013)
3. Hansen, N.: Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In: GECCO Companion, pp. 2389–2396 (2009)
4. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup. Technical report, INRIA (2010)
5. Hansen, N., Müller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* 11(1), 1–18 (2003)
6. Hansen, N., Ostermeier, A.: Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In: International Conference on Evolutionary Computation, pp. 312–317 (1996)
7. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
8. Hansen, N., Ros, R.: Benchmarking a weighted negative covariance matrix update on the BBOB-2010 noisy testbed. In: GECCO 2010: Proceedings of the 12th Annual Conference Comp on Genetic and Evolutionary Computation, pp. 1681–1688. ACM, New York (2010)
9. Hoffmann, F., Holemann, S.: Controlled Model Assisted Evolution Strategy with Adaptive Preselection. In: International Symposium on Evolving Fuzzy Systems, pp. 182–187. IEEE (2006)
10. Igel, C., Hüsken, M.: Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing* 50, 105–123 (2003)
11. Liao, T., Stützle, T.: Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1938–1944. IEEE Press (2013)
12. Loshchilov, I., Schoenauer, M., Sebag, M.: Alternative Restart Strategies for CMA-ES. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 296–305. Springer, Heidelberg (2012)
13. Loshchilov, I., Schoenauer, M., Sebag, M.: Self-Adaptive Surrogate-Assisted Covariance Matrix Adaptation Evolution Strategy. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 321–328. ACM Press (July 2012)
14. Loshchilov, I., Schoenauer, M., Sebag, M.: Intensive Surrogate Model Exploitation in Self-adaptive Surrogate-assisted CMA-ES (saACM-ES). In: Genetic and Evolutionary Computation Conference, pp. 439–446. ACM (2013)
15. Schaul, T.: Comparing natural evolution strategies to bipop-cma-es on noiseless and noisy black-box optimization testbeds. In: Genetic and Evolutionary Computation Conference Companion, pp. 237–244. ACM (2012)
16. Smit, S., Eiben, A.: Beating the ‘world champion’ Evolutionary Algorithm via REVAC Tuning. IEEE Congress on Evolutionary Computation, 1–8 (2010)

Run-Time Parameter Selection and Tuning for Energy Optimization Algorithms

Ingo Mauser¹, Marita Dorscheid², and Hartmut Schneck²

¹ FZI Research Center for Information Technology
76131 Karlsruhe, Germany
mauser@fzi.de

² Karlsruhe Institute of Technology – Institute AIFB
76128 Karlsruhe, Germany
marita.dorscheid@partner.kit.edu, schneck@kit.edu

Abstract. Energy Management Systems (EMS) promise a great potential to enable the sustainable and efficient integration of distributed energy generation from renewable sources by optimization of energy flows. In this paper, we present a run-time selection and meta-evolutionary parameter tuning component for optimization algorithms in EMS and an approach for the distributed application of this component. These have been applied to an existing EMS, which uses an Evolutionary Algorithm. Evaluations of the component in realistic scenarios show reduced run-times with similar or even improved solution quality, while the distributed application reduces the risk of over-confidence and over-tuning.

1 Introduction

Today, the integration of distributed generation, mainly from renewable sources, into energy systems is a major challenge. Techniques, which make loads more flexible, seem to be an efficient way to meet this challenge [10]. One of these techniques is the automated management of energy consumption, generation, and storage in buildings [2]. The differing setups of devices, preferences of the user, and dynamically changing environments at the run-time of an *Energy Management System* (EMS) require an adaptive design of the applied optimization algorithm.

The major contribution of this paper is a run-time parameter selection and meta-evolutionary tuning component for optimization algorithms in EMS. These algorithms are confronted with a wide variety of search and solution spaces, due to the varying scenarios as depicted in Section 2: The scenarios cannot be completely known at design-time of the optimization algorithm, what demands for a concept to adapt the algorithms at run-time of the system. The architecture and mechanisms are described in Section 3. Moreover, a distributed application of the component is presented, which enables collaborative parameter tuning and overcomes the obstacles of over-tuning and over-confidence.

Many approaches to energy management use linear or mixed integer linear programming for optimization [6,8,11]. These systems use an *a priori* formulation of the problem instances that have to be solved. It is assumed that building

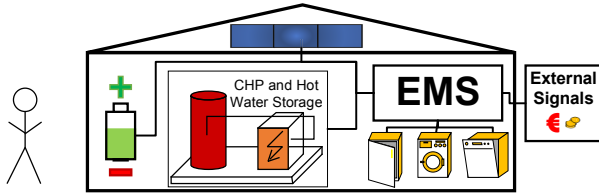


Fig. 1. Smart building scenario with EMS

equipment, user preferences concerning device interaction, and external circumstances form an *a priori* known scheduling problem, which can then be solved by a state-of-the-art system. The broad applicability of EMS in many different buildings is necessary for the integration of distributed energy resources. Thus, the system and its optimization algorithm should be executable on low-power computers with limited system resources. In this context, the running time of the optimization algorithm is crucial, because frequent rescheduling is quite likely. Therefore, the concept of parameter selection and tuning, and its distributed application are applied to an EMS that uses an *Evolutionary Algorithm* (EA) with dynamic formulation of the problem instances at run-time.

Realistic experimental setups for the parameter selection and tuning component and its distributed application are described in Section 4. Results show that problem-specific parameter choices decrease the number of evaluations while keeping or even improving the solution quality in the investigated scenarios.

2 Energy Management Scenario

The focused scenario (see Fig. 1) is the energy management and optimization in smart buildings, which use intelligent energy consuming devices and decentralized energy generation, like photovoltaic systems and combined heat and power plants (CHP). Additionally, batteries or hot water tanks can be installed and controlled, which decouple generation and consumption.

Different devices offer distinct capabilities in terms of influencing the electrical load shape and the overall energy profile of the building. The generators may cause feed-in into the grid or charging of the storage systems. Some energy consuming devices, e.g., household appliances, can be delayed in their operation or interrupted at certain points in their operation cycle. These capabilities allow for a flexible planning of the electrical load by an optimization algorithm.

Load and time variable energy tariffs mirror the external conditions in the grid and on markets. The configuration of the smart building, capabilities of the devices, the variable tariffs as well as goals and preferences of the user form the problem instances in the energy management scenario that the respective optimization algorithm is confronted with. These problem instances are not completely known at the design-time of optimization algorithm. Moreover, all of these aspects vary dynamically over the the run-time of the EMS. These factors call for an adaptive concept for algorithm design.

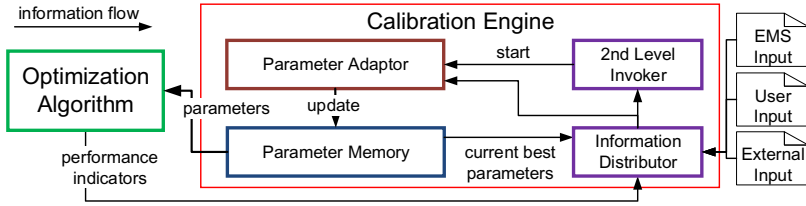


Fig. 2. Calibration Engine: Overview

3 Parameter Selection and Tuning

The *a priori* unknown factors in the energy management scenario may vary the search and solution spaces dramatically. Thus, the configuration of the respective optimization algorithms should be realized at the run-time of the EMS, i.e., after the installation process at the building. This can be realized using an additional component in the EMS.

3.1 Calibration Engine: Architecture

The EMS, which has a configurable *Optimization Algorithm*, is enhanced by an additional component, the *Calibration Engine* (see Fig. 2) that provides the required run-time adaptivity of parameter settings to the algorithm.

The Calibration Engine has two major modules: The first module is the *Parameter Memory* that stores parameter settings for known energy management scenarios. These are stored according to the devices and their respective capabilities in the smart building. Moreover, user's goals and external conditions are part of the storage schema. The second module is the *Parameter Adaptor* that is supposed to find better parameter settings for the concrete problem instances which occur in different scenarios. Additionally, the Calibration Engine has two functionally oriented parts: Firstly, the *Information Distributor* that manages the necessary information, which includes objectives of the user and external signals. Secondly, the *2nd Level Invoker* that determines whether an adaptation of the algorithm's parameters is necessary.

The Optimization Algorithm gets suitable parameter settings at run-time of the EMS by the Parameter Memory. These settings have already been adapted to scenarios that are similar to the current one in terms of the concrete smart building scenario. The 2nd Level Invoker invokes the Parameter Adaptor, when the scenario changes, e.g., due to the installation of new devices or novel user objectives. Then, the Parameter Adaptor systematically tries to find better parameter settings. It therefore uses a *Simulation Model* of the EMS, which is operating in the real-world smart building, to evaluate a parameter setting. By that, the evaluation process does not affect the productive system (comp. Fig. 3). At the end of parameter tuning process, the Parameter Adaptor updates the Parameter Memory with new parameter settings for the current scenario.

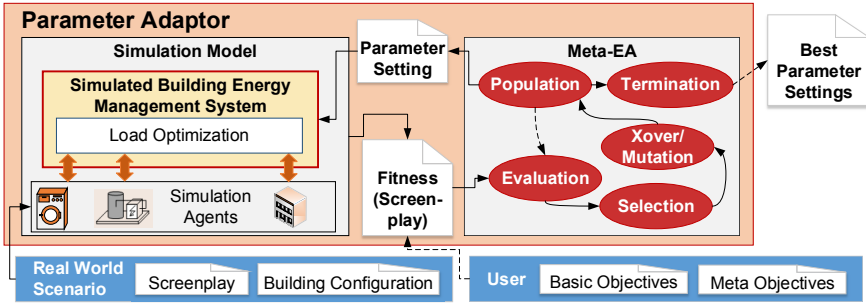


Fig. 3. Structural overview of parameter adaption process and data

3.2 Parameter Adaptor and Process of Parameter Tuning

Parameter tuning requires a systematic selection and evaluation of different parameter settings. It has to be ensured that the search space of different combinations is at the same time explored as already good solutions are exploited by some local search. Several approaches to parameter tuning have already shown good results in improving the solution quality of meta-heuristic algorithms. These include, *Iterated Local Search* algorithms [9], which utilize iterative mechanisms of local search and acceptance criteria, and *Sequential Parameter Optimization* [3], which is based on statistically derived models of the search space.

The mechanism presented hereafter, which is used for the generation of the parameter settings, extends a meta-evolutionary parameter tuning approach by [7] and adapts it to optimization algorithms in EMS. Its main advantage is the simple and flexible structure that allows for a distributed, parallel evaluation of candidate solutions in the domain of energy management. Additionally, the EA can be used twice: If an EA is already used in the load optimization, which has been presented in [1], this EA can be re-used for the parameter tuning. This simplifies the system design of an EMS and reduces its complexity.

Parameter tuning by the Parameter Adaptor (see Fig. 3) for a certain building and situation requires extensive information about the concrete real world scenario and the user objectives. The scenario consists of the *Building Configuration*, i.e., the available devices and their specific capabilities, as well as the *Screenplay* that is the recorded pattern of user behavior and interaction, device usage, and further relevant information. The objectives of the Meta-EA have to mirror the ones in the real system, e.g., overall cost minimization, but can also take additional sub-objectives into account, e.g., reduction of evaluations while keeping a certain solution quality. Candidate parameter settings are represented as real-valued genes of the meta-individuals of an EA. The evaluation of a meta-individual is realized by loading the Simulation Model, which consists of replica of the productive real world EMS, the building and the devices. This model is necessary to simulate the usage of energy and to calculate the fitness of parameter settings. The optimization is applied to a certain Screenplay, which, for comparability reasons, has to be the same for each individual evaluated in the Meta-EA.

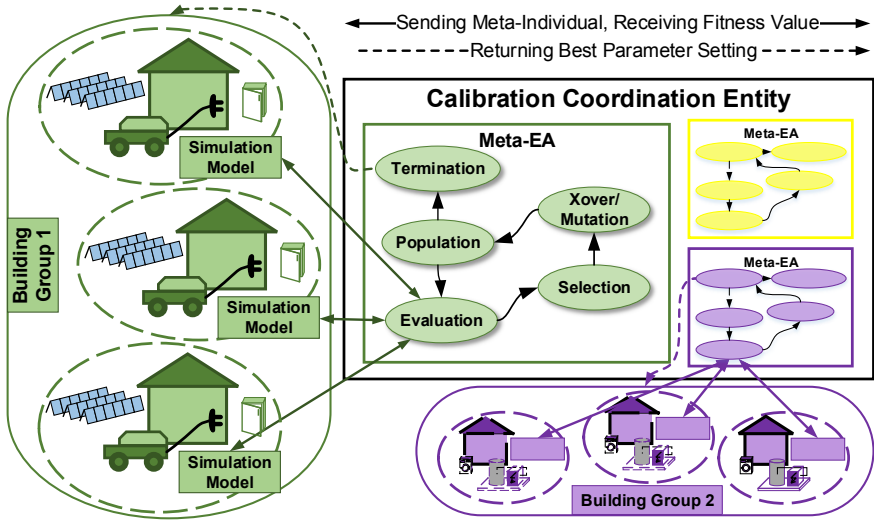


Fig. 4. Calibration Coordination Entity with different groups of buildings

3.3 Distributed Evaluation

As the target platforms of the EMS in real-world scenarios are low-power computers with limited resources, the described parameter tuning process can only be performed in the otherwise idle time of the system. Therefore, the EMS should take advantage of a wider range of information from similar buildings and a distributed evaluation of parameter settings as follows.

The working of the parameter tuning is now enhanced to a distributed approach using multiple buildings. Similar buildings, i.e. similar scenarios, are grouped according to their equipment, objectives, energy consumption, and typical behavior of users. These grouping criteria are called *characteristic parameters*. The *Calibration Coordination Entity* (CCE) shown in Fig. 4 executes a Meta-EA for every group of similar buildings. The evaluation of a parameter setting, which is represented by a meta-individual, is performed in a distributed manner in the EMS of the buildings. There, the simulation model described in the previous section calculates a local fitness by applying the parameter setting from the CCE to the optimization using the locally recorded Screenplay. The resulting fitness is communicated to the CCE, where it is averaged over all buildings of one group. The fitness of a meta-individual is thus the averaged fitness of similar buildings.

At the end of the distributed parameter adaption process, the found parameter settings are more closely adapted to the characteristic parameters than to a specific Screenplay. Therefore, they are better applicable throughout all buildings of their respective group and their typical, not their one-time behavior represented in a single Screenplay. Additionally, this distributed approach is sensitive towards data privacy, since the Screenplays, which reflect very intimate data of the users, do not have to be exchanged with an instance outside of the building.

4 Experimental Setup

In order to investigate the parameter selection and tuning component for optimization algorithms in EMS and the distributed application of this component, they were implemented for an EMS that is already in use in two exemplary smart buildings¹.

This EMS uses EA for optimization purposes [1]. The parameters that have to be selected and tuned for this EA are crossover and mutation probability. Additionally, the ratio of population size and number of generations are investigated to reduce the running time of the algorithm. Thus, it supports the execution of the EMS on low-power computers with limited system resources.

In the following section, the energy management scenarios for the experiments, the *test scenarios*, are described. From these test scenarios, Screenplays were generated, which represent the problem instances occurring in the concrete energy management scenario. Afterward, the experiments are depicted in Section 4.2.

4.1 Test scenarios

There are a few major factors that determine an energy management scenario: First of all, there is the configuration of the smart building concerning the available devices. Furthermore, the capabilities of these devices in terms of influencing the load shape by optimization are distinct.

Five basic appliances and a combined heat and power plant (CHP) with different capabilities form the basis for the test scenarios in this paper (see Tab. 1a). A *non-delayable* device's operation always starts immediately when used and thus does not have to be optimized, though it still has to be considered. In contrast, the operation time of a *delayable* device may be shifted by the optimization, while complying with the user's preferences. Additionally, an *interruptible* device offers the capability to be paused at certain points in its operation cycle.

The capabilities of the CHP, which are both connected to a hot water storage system, are differentiated into *non-controllable* and *controllable* by the EMS. Non-controllable stands for a thermal management of the CHP, meaning that it is switched on and off according only to the thermal demands in the building. In contrast, the controllable CHP can be switched on, whenever capacity is left in the storage. Of course, the thermal demand and local limitations of the storage system, e.g., the minimum threshold temperature, still have to be respected.

Another determining factor of energy management scenarios is the user. On the one hand, the overall goal of the user is cost minimization. On the other hand, the user behavior has to be considered. Her preferences are represented by the electrical demand, e.g., when the appliances are used and how long they may be delayed if possible. The maximum delay is set to eight hours across all test scenarios. The user's thermal demand is modeled as a 5-person-household.

The last determining factors are the external conditions. They are mirrored by a time-variable energy tariff based upon a market simulation as described in [4].

¹ KIT Energy Smart Home Lab <http://www.izeus.kit.edu/english/> and FZI House of Living Labs <http://www.fzi.de/en/fzi-house-of-living-labs/>

Table 1. Devices and configurations of households used in simulation
 (a) Different devices (b) Configurations of households

Name	Device	Capability	Configuration	Devices
D1N	Hob	Non-delayable	H0	D1N, D2N, D3N, D4N, D5N, CHP0
D2N	Dishwasher	Non-delayable	H1	D1N, D2D, D3N, D4D, D5D, CHP1
D2D	Dishwasher	Delayable		
D3N	Oven	Non-delayable	H2	D1N, D2D, D4N, D4I, D5D, CHP1
D4N	Dryer	Non-delayable		
D4D	Dryer	Delayable	H3	D1N, D2D, D3N, D4D, D5D, CHP0
D4I	Dryer	Interruptible		
D5N	Washing machine	Non-delayable	H4	D1N, D2D, D3N, D4I, D5D, CHP0
D5D	Washing machine	Delayable		
CHP0	CHP	Non-controllable		
CHP1	CHP	Controllable		

The tariff changes every hour and ranges from 3 to 39 ct/kWh with a mean value of 24 ct/kWh , respectively. Moreover, a load limit is set to 3 kW across all H0–H4. When the power limit is violated, the amount of energy consumed above the limit is penalized by a doubling of electricity costs. The decentralized generator, the CHP, produces electrical and thermal energy by the consumption of natural gas. The gas price of 6 ct/kWh_{th} ² is constant. If the electricity generation exceeds the current consumption in the building, the difference is fed into the grid, receiving constant feed-in compensations of 5 ct/kWh_{el} .

From the devices above, five configurations for smart buildings had been assembled (see Tab. 1b). These configurations are furthermore referred to as *households* H0–H4. The problem instances for the parameter tuning process—the Screenplays—were generated according to typical times of use for each household. To reflect differing thermal demands that effect the optimization of the CHP, ten Screenplays per household are located in January (winter) and ten Screenplays are located in July (summer).

4.2 Experiments

The Calibration Engine and its distributed application are confronted with a set of experiments that are based on the test scenarios described in the last section. The reference parameter setting of the EA consists of a crossover probability of 0.7, a mutation probability of 0.1, a binary tournament selection of parents, a single-point-crossover with two offspring and a bit-flip-mutation using an elitist (μ, λ) -strategy with a rank based survivor selection. The stopping criteria is a maximum number of evaluations, determined by varying numbers of generations and individuals. The Meta-EA has been set up as follows: ten generations of 24 individuals, SBXCrossover [5] with a probability of 0.7, and polynomial mutation with a probability of 0.3, both with a distribution index of 20.

The fitness of a certain parameter setting is evaluated by the calculation of the *average electricity costs* (AEC). AEC are given by the average price per kWh

² Due to the constant degree of efficiency, the price is non-varying over kWh_{el} , too.

that results from consumption from the grid as well as from the generation by the CHP and its consumption of natural gas.

In the first experiment, the Parameter Adaptor is confronted with one problem instance (Screenplay) per household. Afterward, the found settings are applied to all ten corresponding Screenplays. This proceeding should simulate the usage of the Parameter Memory in a single building with EMS. Moreover, risks of over-tuning and over-confidence should be identifiable. In the second and third experiment, the Calibration Engine is used in the distributed application. The Parameter Adaptor evaluates the fitness of the found parameter setting according to the averaged resulting fitness of three respective five Screenplays per household. Afterward, the parameter settings are applied to all ten corresponding problem instances. This experiment investigates the potential of reduction of the risk of over-tuning and over-confidence by the distributed approach.

5 Results and Discussion

A comparison of two exemplary fitness landscapes in Fig. 5 visualizes the seasonal influences on the parameters, which mainly result from seasonally different device usage and thermal demand. The result of the Meta-EA for the configurations H1 and H2, different maximum number of evaluations and included Screenplays with corresponding outcomes of Avg. EC are shown in Fig. 6. "Average" in this case means that the results of all ten Screenplays of buildings were averaged.

The results show that on the one hand the parameter tuning is able to exploit considerable potentials of optimization. On the other hand, the advantage induced by adapting the parameter settings is remarkable due to the possibility to reduce evaluations, while resulting in the same level of solution quality compared to the run with 10,000 evaluations and the initial parameters. This means that individual parameters can successfully reduce execution time of the EA in the EMS, without worsen its results. Moreover, the results also show

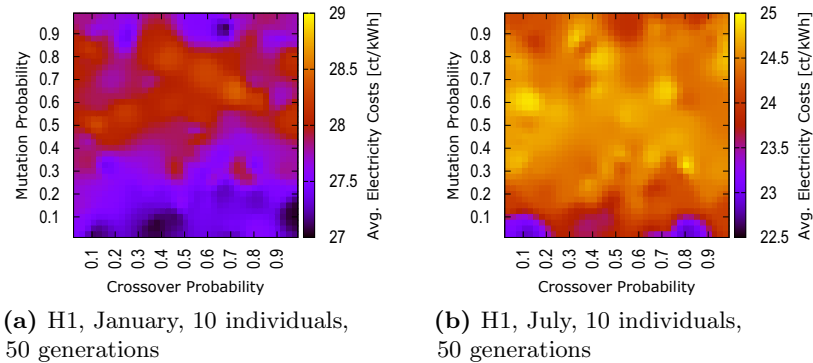


Fig. 5. Fitness landscapes of January and July showing distinct areas of good parameter settings

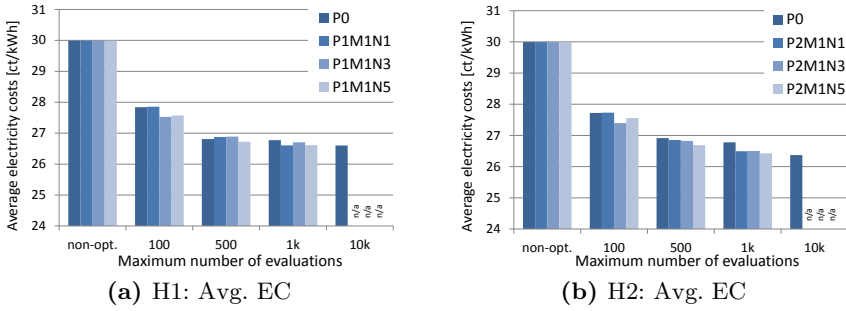


Fig. 6. Simulation results of H1 and H2 with default (P0) and optimized parameter settings (P1, P2) using N Screenplays in January (M1)

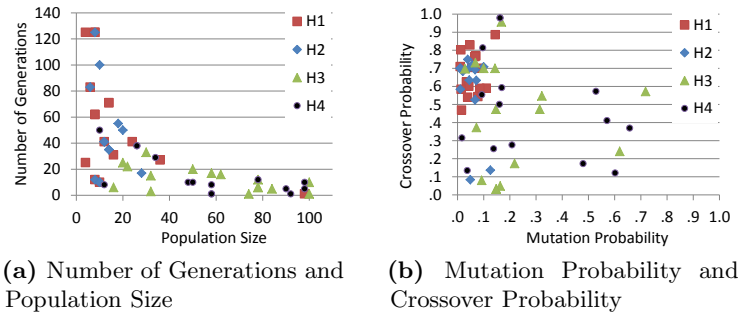


Fig. 7. Resulting parameter settings for households H1 – H4

that the distributed optimization approach tends to result in better parameter settings, which are better applicable throughout the ten Screenplays of similar households.

Resulting parameter settings differ across the building configurations. Fig. 7 visualizes optimized parameter settings for all test scenarios H1 – H4. Simulation setups with a controllable CHP (H1 and H2) tend to have lower ratios of population size to number of generations (see Fig. 7a) and more clustered parameter combinations of crossover and mutation probability (see Fig. 7b), whereas setups with a non-controllable CHP (H3 and H4) show a larger spreading.

The potential of parameter tuning has been shown across different experimental setups. The Calibration Engine was able to exploit potentials, although it sometimes produces parameters settings with worse results when applied to all corresponding Screenplays than the initial settings. Nevertheless, the Calibration Coordination Entity was able to tackle this issue by averaging the fitness of parameter settings. This is important, because Screenplays always represent past behavior of households which is likely to never happen exactly the same again. Therefore, a better fitness with Screenplays of other similar households will lead to better results within the same household in a similar future month.

6 Summary and Outlook

This paper presented a run-time parameter selection and tuning component for optimization algorithms in Energy Management Systems and an approach to a distributed application of this component. An implementation for an EMS, which uses a run-time formulation of the problem instances and an EA to optimize them, is presented. In this context, the component has been tested and has shown potential to decrease the average electricity costs while reducing the running time per optimization process. The parameter tuning has reacted sensitively to different configurations of devices, capabilities of devices and user preferences.

It has been shown that parameter tuning in the domain of EMS and thus enhances a broad applicability of EMS at the level of buildings. Future work shall further validate the component, also taking into account more parameters and other optimization algorithms. Moreover, it shall be applied to other real-world implementations of EMS.

References

1. Allerding, F., Premm, M., Shukla, P.K., Schmeck, H.: Electrical Load Management in Smart Homes Using Evolutionary Algorithms. In: Hao, J.-K., Middendorf, M. (eds.) *EvoCOP 2012*. LNCS, vol. 7245, pp. 99–110. Springer, Heidelberg (2012)
2. Allerding, F., Schmeck, H.: Organic smart home: architecture for energy management in intelligent buildings. In: *Proceedings of the 2011 Workshop on Organic Computing*, pp. 67–76 (2011)
3. Bartz-Beielstein, T., Lasarczyk, C.W., Preuß, M.: Sequential parameter optimization. In: *The 2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 773–780 (2005)
4. Dallinger, D.: The contribution of vehicle-to-grid to balance fluctuating generation: Comparing different battery ageing approaches. Tech. rep., Working Paper Sustainability and Innovation (2013)
5. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. *Complex Systems* 9, 1–34 (1994)
6. Di Giorgio, A., Pimpinella, L.: An event driven Smart Home Controller enabling consumer economic saving and automated Demand Side Management. *Applied Energy* 96, 92–103 (2012)
7. Freisleben, B., Härtfelder, M.: Optimization of genetic algorithms by genetic algorithms. In: *Artificial Neural Nets and Genetic Algorithms*, pp. 392–399. Springer (1993)
8. Ha, D.L., Joumaa, H., Ploix, S., Jacomino, M.: An optimal approach for electrical management problem in dwellings. *Energy and Buildings* 45, 1–14 (2012)
9. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research* 36(1), 267–306 (2009)
10. Palensky, P., Dietrich, D.: Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE Transactions on Industrial Informatics* 7(3), 381–388 (2011)
11. Zhang, D., Papageorgiou, L.G., Samsatli, N.J., Shah, N.: Optimal scheduling of smart homes energy consumption with microgrid. In: *ENERGY 2011*, pp. 70–75 (2011)

Towards a Method for Automatic Algorithm Configuration: A Design Evaluation Using Tuners

Elizabeth Montero and María-Cristina Riff*

Department of Computer Science
Universidad Técnica Federico Santa María
Valparaíso, Chile
{Elizabeth.Montero, Maria-Cristina.Riff}@inf.utfsm.cl

Abstract. Metaheuristic design is an incremental and difficult task. It is usually iterative and requires several evaluations of the code to obtain an algorithm with good performance. In this work, we analyse the design of metaheuristics by detecting components which are strictly necessary to obtain a good performance (in term of solutions quality). We use a collective strategy where the information generated by a tuner is used to detect the components usefulness. We evaluate this strategy with two well-known tuners EVOCA and I-RACE to analyse which one is more suitable and provides better results to make this components detection. The goal is to help the designer either to evaluate during the design process different options of the code or to simplify her/his final code without a loss in the quality of the solutions.

Keywords: Automated algorithm tuning, automated algorithm configuration, metaheuristics.

1 Introduction

In order to obtain a metaheuristic with good performance, we have to make several design decisions. The design process is usually iterative, and at each step, the involved components must be evaluated. On the other hand, the final code of the metaheuristic can be extremely complex. Thus, analysing and understanding its results becomes difficult, and this is often a very time consuming task. Unexperienced designers usually tend to include more and more components during the iterative design process of a metaheuristic, without evaluating the usefulness of previously incorporated ones. We propose to have intermediate refining steps, during the design process, in order to help the designer to obtain a simpler design with similar performance. Our motivation is to assist the designer to make good decisions in order to produce an efficient metaheuristic (e.g. in terms of the solutions quality). In this paper, we study the information produced by the tuners, and compare their ability in helping the designer. We briefly revise published works related to this subject in section 2.

* This work is supported by the Fondecyt project 1120781 and Postdoctoral Fondecyt project 3130754. María Cristina Riff is partially supported by the Centro Científico Tecnológico de Valparaíso (CCTVal) No. FB0821.

In section 3, we introduce the general problem when designing metaheuristics and give a general idea on how design should be achieved. As a particular instance of this idea, we show how we can use the Evolutionary Calibrator (EVOCA) [13] and I-RACE [3] tuners that can work with categorical and numerical parameters [10]. We provide details on how to use a collective strategy based on these tuners in section 4. To evaluate our proposal, we use two well-known metaheuristics: A genetic algorithm that solves the NK-landscapes problems (NK-GA) [11], and a more complex one which is an artificial immune system algorithm that solves multiobjective problems (MOAIS-HV) [12]. These metaheuristics have already shown good performance to solve these kind of problems. Moreover, MOAIS-HV has a quite complex implementation and uses several components with several explicit and implicit parameters (not detailed in its description), which makes it a very interesting case for our work. We give brief description of both NK-GA and MOAIS-HV in section 5, as well as the results of a set of experiments to evaluate our solution. This section also provide a statistical comparison between the results obtained using EVOCA and I-RACE tuners. It is important to remark that our goal is not to find the best solution to the problem to be solved, but to focus on the way to detect the components that are strictly required from the ones that are unnecessary. Finally, we present the conclusions and future work in section 6. The contributions of this paper are: A general iterative method to select the best components and simplify the metaheuristic code, a comparison of the suitability of the EVOCA and I-RACE tuners to help into designing by refining of metaheuristics, and an in-depth analysis of the data generated by both tuners which can be used to make better decisions during the design process.

2 Related Work

Our approach shares much of its motivation with existing work on automated parameter tuning and algorithm configuration [5]. In automated parameter tuning the design space is defined by an algorithm whose behavior is controlled by a set of parameters, and the task is to find performance-optimizing settings of these parameters. For this, various methods can be used ranging from well-known numerical optimization procedures such as gradient-free CMA-ES algorithm [7] to discrete approaches based on experimental design methods [2], response-surface models [1] or stochastic local search procedures [8]. In automated algorithm configuration, the design space is defined by an algorithm scheme that contains a number of instantiable components, along with a discrete set of concrete choices for each of these. It can be mapped as a tuning problem, in which categorical parameters are used to select a set of components to instantiate the given scheme. However, only ParamILS and a genetic programming procedure applied to the configuration of local search algorithms for SAT [6] have been obtained promising results. Our approach is also related to the hyperheuristic methods [4]. When designing hyperheuristics, the goal is to find a good design with sufficient quality, and not to find the best algorithm. It is very useful to tackle real-world problems which must be quickly solved.

3 The Problem When Designing Metaheuristics

We can identify two problems when designing metaheuristics: The On-the-fly design problem and the post-design problem or refining problem. The first one occurs during the design process and is about the decisions to make when building efficient metaheuristics. Some of these decisions are related to the components or procedures to append to the algorithm to improve its performance. The On-the-fly Metaheuristic Design problem (OMD) can be stated as follows: given an intermediate design step, the current code of a metaheuristic M , and a set of candidate components $S = \{C_1, \dots, C_n\}$ to be included in M . The OMD consists in finding a code M' for the metaheuristic using the selected components among the already included and the new candidates, in order to improve the performance of M . Unlike parameter control strategies, OMD problem is focused on how solution tool is constructed by selecting useful components from a set of possible options, like in a hyperheuristic approach.

The second problem occurs during post-design of the metaheuristic when we are interested in simplifying its code without performance loss.

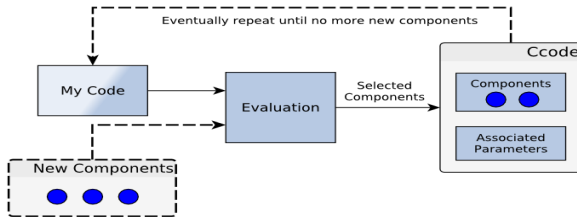


Fig. 1. Illustration of the designing problems

The Refining Metaheuristic Design problem (RMD) can be stated as follows: given M , a target algorithm or metaheuristic, a set of parameters for the algorithm and a set of input data. The RMD consists in finding a reduced code for the metaheuristic which gives, at least, the same performance with the same input data than M . Figure 1 illustrates both problems. For the OMD problem (dotted lines), when we are building My Code we evaluate the inclusion of new components to improve its performance. After this evaluation a current code Ccode is obtained, which can follow a new procedure of inclusion of new components. For the RMD problem when the code is finished it follows an evaluation step in order to identify, when this is possible, an alternative code which uses a reduced number of its components but has at least the same performance.

4 Strategy to Use Tuners for Designing Metaheuristics

Collective Strategy : The key idea in the collective strategy is to generate a competition between the components that we are evaluating. The role of the tuner is to help us in the decision process. In this work we map RMD and OMD to a configuration problem, where new parameters are introduced to the algorithm in order to identify alternative designs for M . More formally,

Definition 1. Given a metaheuristic code M , an instance of the problems consists in a 6-tuple $P = (M, S, \Theta, \Pi, \kappa_{max}, M')$, where S is the set of new binary parameters introduced in M that allows to either turn on or off some components. M' is the modified version of M that includes S . Θ is the configurations space for M' . Π is the set of input problem instances, $g(\theta, \Pi)$ is a function that computes the expected gain (e.g., the quality of the solutions) of running M' using instance $\pi \in \Pi$ when using configuration θ . κ_{max} is a time out after which all instances of M' will be terminated if they are still running.

Any configuration $\theta \in \Theta$ is a candidate configuration of P . The gain of a candidate configuration θ is given by:

$$G_P(\theta) = \text{mean}_{\pi \in \Pi}(g(\theta, \pi)) \quad (1)$$

This definition considers the mean of gain induced by $g(\theta, \pi)$, but any other statistic could be used instead (e.g. median, variance). Given G^M , the gain of metaheuristic M using its best parameter configuration, an alternative code defined by the configuration θ^* has a value $G_P(\theta)$ such that:

$$G_P(\theta^*) \geq G^M \quad (2)$$

In this definition for the OMD problem M is the current code and M' also includes the alternative components to be evaluated. For the RMD M is the final metaheuristic code. We define the **collective strategy** as the evaluation of the metaheuristic M' using different parameter configurations. For this strategy, we use the tuner to obtain the P_1, P_2, \dots, P_k binary values that indicate which of the k components must be turned on in the M' code. The evaluation of the performance of the algorithm with its best set of parameters values is used to decide whether or not the algorithm can be modified.

Definition 2. Given M a metaheuristic with a performance G^M and M_1, \dots, M_l alternative algorithms with performances $G_{M_1}(\theta_1), \dots, G_{M_l}(\theta_l)$ for a maximization problem. M_i belongs to the **set of alternative designs** S_d if and only if $G_{M_i}(\theta_i) \geq G^M$. We define M_i as the **best alternative design** such that $G_{M_i}(\theta_i) \geq G_{M_j}(\theta_j), \forall M_i, M_j \in S_d, i \neq j$

When two or more best alternative designs are identified, the decision will be to use the simplest one. When we use the collective strategy during the post-design we are looking for a possible alternative code that allows the metaheuristic M' to solve the problems as M does with at least the same performance, but using a reduced number of its initial components. Given its stochastic nature, it is noteworthy to mention that the refined algorithm could show better performance than the initial one. When we use the collective strategy during the design we are looking for a code M' which has better performance than M and is composed by a new set of components.

4.1 How to Use Tuners during the Design Process

The designer usually follows an incremental procedure for the components selection. At the beginning, the designer has a first set of components that he/she

believes to be some good candidates to include in the code. The typical question is therefore: Which of these components are more suitable to be included in my code?. In other words, which of these components do have the best performance to solve my problem. Using the collective strategy, the designer can determine which are the best code alternatives according to the information provided by the calibrator when using these components. Let's call *CCode* the current code, which corresponds to the best one obtained by this evaluation. Then, the designer can add to the *CCode* a set of new components that he/she thinks they could improve the metaheuristic. A new evaluation is made with the collective strategy to determine which components among the previously selected and the new ones allows the algorithm to have the best performance, that is which is the best alternative for the new *CCode*. Note that all the components can be turn on or off during the evaluation and therefore, previous selected ones may be discarded at this step. This is mainly because some new components could do the same search than previous selected ones, but more efficiently. The same process is repeated until the designer does not have more components to add. The final *CCode* is then the best one determined by the collective strategy. Figure 2 shows an example where 4 components (C_1, C_2, C_3, C_4) are initially considered to be included into the code. The information provided by the calibrator allows to select a *CCode* that includes (C_1, C_2) as the best alternative. Then, in a second step, the designer considers to include components (C_5, C_6, C_7) into the code. The evaluation of the tuner suggests to include components (C_6, C_7), but in this case to discard C_1 which has been selected in the previous step. The design process continues until obtaining a Final Code of high quality. Many other design options can be considered. For instance, the inclusion of a component associated to a particular method could be easily evaluated together. Moreover, it is also possible to consider the inclusion of a component that was previously discarded before and to evaluate its inclusion at the current time of the design process.

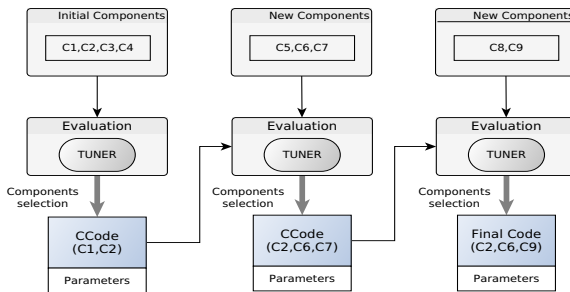


Fig. 2. Example for using the Collective Strategy

5 Experiments

The purpose of these experiments is to analyse different scenarios of using the collective strategy. Any tuning method able to calibrate categorical parameters can

be used by our framework: sampling, model-based, screening or meta-evolutionary methods [5]. In our study we consider the following I-RACE [2] and the EVOCA [13] tuners.

I-RACE or Iterated F-Race: Iterated F-Race¹. is an iterative version of F-Race algorithm [2]. At each iteration, Iterated F-Race uses a number of surviving candidate parameter configurations to bias the sample of new candidate configurations. Iterated F-Race follows the framework of model-based search: (1) construct a candidate solution based on some probability model; (2) evaluate all candidates and (3) update the probability model of biasing the next sample.

The EVolutionary CALibrator: EVOCA² is itself an evolutionary algorithm that works with a population of parameter configurations. It uses two operators. Wheel-crossover constructs one child from the whole population. The child replaces the worst individual in the current population. The mutation operator is a hill climbing procedure. The child generated by mutation replaces the second worst individual in the current population, when finding a better individual.

5.1 Experiments with NK-GA

For these experiments we use a genetic algorithm that solves unrestricted NK landscape problems [9], proposed in [11].

NK-GA: This genetic algorithm (GA) evolves a population of fixed-length binary strings. New solutions are created by applying variation operators to the population of selected solutions. The algorithm has three genetic operators bit-flip mutation, uniform crossover and two-point crossover.

Test Suite: We perform our experiments using a set of unrestricted NK landscape instances with k ranging from $k = 2$ to $k = 6$. A total number of 15 problem categories are considered. The minimization of the number of evaluations required by the genetic algorithm to solve all the instances is considered as a criteria to evaluate the parameter configuration. We consider a budget of 3500 runs for both tuners. We focus here on the use of the three genetic operators: Uniform crossover(U), two-points crossover (T) and bit-flip mutation (M). We run each tuner 20 times and we show the best 5 results as well as the execution time in table 1. For both tuners, the set of alternative designs S_d includes combinations that use less components than the original algorithm. They solve all the instances. Thus, using both tuners we can identify that both crossover components are not necessary to the algorithm for solving the 15 categories of the problem. We can also remark that using only one or two components instead of the three initial ones neither implies a significant increase of the number of evaluations nor of the execution time. In conclusion, both the uniform crossover and the two-points crossover are not strictly necessary to the algorithm performance. Similar results were obtained using a reduced number of NK-GA evaluations.

¹ I-RACE is available from *CRAN*, <http://cran.r-project.org>

² EVOCA is available in our website comet.informaticae.org

Table 1. Algorithms selected by EVOCA and I-RACE

EVOCA					I-RACE				
	Average solved	Average success	Average Evaluations	Average Time [s]		Average solved	Average success	Average Evaluations	Average Time [s]
UM	15	100	116.2	0.01	UTM	15	100	106.3	0.01
M	15	100	121.0	0.01	UTM	15	100	113.1	0.01
UTM	15	100	125.6	0.01	UTM	15	100	122.5	0.01
TM	15	100	131.4	0.01	M	15	100	137.4	0.01
UM	15	100	144.5	0.01	UM	15	100	141.1	0.01

5.2 Experiments with MOAIS-HV

We now use the MOAIS-HV algorithm proposed in [12] to solve multiobjective optimization problems, to show the effectiveness of our proposal.

MOAIS-HV: The main idea of MOAIS-HV is to maintain an online population of antigens and antibodies. In this case antigens are considered to be good quality solutions and antibodies are the bad ones. The antigens are cloned and a mutation operator is applied. The mutated clones and the best antigens found are merged and the size of the main population is maintained by discarding individuals that contribute the least to maximize the hypervolume.

Test Suite: In this case, we want to analyze the design of the MOAIS-HV's hypermutation process. In MOAIS-HV, hypermutation is applied to each variable according to a probability value. Each time a variable has to be mutated, the probability is recomputed. This probability indicates a trade-off between the Global Gaussian (G) and the Local Gaussian (L). The probability changes as the algorithm goes on and the hypermutation will perform more/less local searches. The changing probability criteria is based on the computation of a complex formula that uses explicit and some implicit values which are obtained by other procedures on the code.

The goal of our experiments is to analyse if the algorithm really needs this complex hypermutation process. We evaluate the use of the Global and the Local mutation to obtain a good performance. We also evaluate if it is really required to change the probability of using L during the search. For our experiments we use well-known 2-objectives standard functions: zdt1-zdt4, zdt6 and 3 objectives: dtlz1-dtlz7. These are the same functions used by the authors to introduce MOAIS-HV. For the 2-objectives functions, we consider a population of size 100 and a maximum of 200 iterations. For the 3-objectives functions, the population size is 200, and there is a maximum of 500 iterations. The test consists in the maximization of the hypervolume of the previously mentioned functions. This maximization is used to evaluate each configuration. Source code of MOAIS-HV is also available in our website³. In this experiment, we want to evaluate both the inclusion of the three components in the code: Global mutation(G),

³ comet.informaticae.org

Table 2. Algorithms and their performance found by EVOCA and I-RACE

EVOCA			I-RACE		
<i>Algorithm</i>	Average Hypervolume	Average Time [s]	<i>Algorithm</i>	Average Hypervolume	Average Time [s]
LR	0.790047	1.29	GR	0.777034	1.73
GL	0.775806	1.32	GLR	0.789478	1.81
LR	0.786416	1.71	GLR	0.787906	1.72
LR	0.781908	1.82	GLR	0.789524	1.80
GLR	0.781233	1.80	GLR	0.784471	1.43
GLR	0.784627	1.69	GLR	0.789200	1.64
GLR	0.783951	1.68	GLR	0.789115	1.67
GLR	0.785932	1.69	GLR	0.789502	1.63
GLR	0.784209	1.68	GLR	0.785856	1.64
LR	0.787449	1.65	GLR	0.789323	1.68
GLR	0.785943	1.69	G	0.780221	1.65
GLR	0.784256	1.67	GLR	0.780044	1.67
LR	0.790048	1.66	GLR	0.786530	1.67
GL	0.784752	1.67	GLR	0.788129	1.64
LR	0.790048	1.66	GLR	0.788683	1.67

Local mutation(L) and Random mutation(R) and the method to apply them. Thus, we initially consider a code where the mutation components have the same fixed and equally probability to be applied. The tuning process considers all the functions together. When using EVOCA, the relative difference to the best solution found was used to compare the performance of different test functions. For both tuners, 12000 runs are set as maximum budget.

Analysis: Table 2 shows the algorithms found by EVOCA and I-RACE, their execution times and the performance measured in their 15 executions. The performance is the average hypervolume of 50 runs with different seeds. Differences in performance when the same algorithms were selected are due to differences in the parameters values tuned at each execution. For EVOCA, the best code uses the Local and Random Mutation, and a fixed probability value. For I-RACE, the best option uses the three mutations, also with a fixed probability value. This code also improves the average performance of the MOAIS-HV, but less significantly than the algorithm selected by EVOCA.

Statistical Evaluation: Table 3 shows the statistical analysis indicators obtained using Wilcoxon test. Considering p-value=0.05, the algorithms identified by both, EVOCA and I-RACE, outperform the original algorithm. Moreover, EVOCA's algorithm in 364 cases is better than original MOAIS-HV and I-RACE's algorithm in 336. Table 4 shows the performance obtained by the algorithm when solving each function. We observe that both, the original algorithm and the algorithm found by I-RACE, obtained the best performance in 3 functions, and the algorithm defined by EVOCA shows a better performance in 6 functions. In terms of the execution time, the algorithm selected using EVOCA and the one using I-RACE are similar. In both cases the code obtained is much simpler than the

Table 3. Wilcoxon ranks

Ranks	MOAIS-HV - EVOCA			MOAIS-HV - I-RACE		
	N	Mean Rank	Sum of Ranks	N	Mean Rank	Sum of Ranks
Negative Ranks	364	322.80	117497.5	336	346.90	116559.0
Positive Ranks	179	168.71	30198.50	208	152.31	31681.00
Ties	57			56		
Total	600			600		
Statistics						
Z						-11.93
Asymp. Sig. (2-tailed)						0.00

Table 4. Performance comparison

Function	MOAIS-HV	EVOCA's Algorithm	I-RACE's Algorithm
zdt1	0.871372	0.868736	0.870774
zdt2	0.538028	0.535536	0.537482
zdt3	1.328516	1.326314	1.327164
zdt4	0.814616	0.825334	0.813102
zdt6	0.504312	0.504320	0.504300
dtlz1	0.314374	0.316040	0.316274
dtlz2	0.744758	0.748010	0.747234
dtlz3	0.670666	0.735176	0.741356
dtlz4	0.741330	0.749178	0.749238
dtlz5	0.434040	0.434300	0.431530
dtlz6	0.431980	0.437506	0.436822
dtlz7	1.987018	2.000120	1.999014

original one with a same or best level of performance. There is especially no more complex formula to compute the dynamic probability for the mutations.

6 Conclusions and Future Work

In this work, we have analysed two problems about the metaheuristics design. The On-the-fly metaheuristics design problem (OMD) occurs during the design process and the Refining Metaheuristics Design problem (RMD) which is a post-design problem. The OMD problem, which concerns the selection of the components and methods to include in the code is always present when designing metaheuristics, but the RMD problem does not always necessary and strongly depends on the designer goals. We have compared two well-known tuners to help the designer during the evaluation process, but any tuning method able to work with categorical parameters can be used. We have evaluated the code selected by EVOCA and I-RACE for two metaheuristics: a genetic algorithm (NK-GA) and an artificial immune algorithm (MOAIS-HV). Both tuners have shown to be able to detect the most suitable components, and to produce simpler and efficient algorithms. Moreover, in terms of the performance, there is not a statistical significant difference between their identified algorithms. The results obtained indicate the suitability of both tuners for helping the designer evaluation task

and could be used in the future for including on a framework for automatic configuration algorithms or hyperheuristics.

Acknowledgment. We thank Dr. Carlos Coello for MOAIS-HV code and Mrs. Leslie Pérez for her support with I-RACE implementation.

References

1. Bartz-Beielstein, T.: *Experimental Research in Evolutionary Computation—The New Experimentalism*. Natural Computing Series. Springer (2006)
2. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A Racing Algorithm for Configuring Metaheuristics. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 11–18. Morgan Kaufmann, USA (2002)
3. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and Iterated F-Race: An Overview. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) *Experimental Methods for the Analysis of Optimization Algorithms*, pp. 311–336. Springer, Heidelberg (2010)
4. 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. (eds.) *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, vol. 57, pp. 457–474. Springer, US (2003)
5. Eiben, A.E., Smit, S.K.: Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms. *Swarm and Evolutionary Computation* 1(1), 19–31 (2011)
6. Fukunaga, A.: Automated Discovery of Composite SAT Variable Selection Heuristics. In: *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 641–648 (2002)
7. Hansen, N., Kern, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiño, P., Kabán, A., Schwefel, H.-P. (eds.) *PPSN 2004*. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004)
8. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research* 36, 267–306 (2009)
9. Kauffman, S.A.: *Adaptation on Rugged Fitness Landscapes*. *Lecture Notes in the Sciences of Complexity* 1, 527–618 (1989)
10. Montero, E., Riff, M.C., Neveu, B.: A Beginner’s Guide to Tuning Methods. *Applied Soft Computing* 17(0), 39–51 (2014)
11. Pelikan, M.: Analysis of Estimation of Distribution Algorithms and Genetic Algorithms on NK landscapes. In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO*, pp. 1033–1040. ACM, USA (2008)
12. Pierrard, T., Coello Coello, C.A.: A Multi-Objective Artificial Immune System Based on Hypervolume. In: Coello Coello, C.A., Greensmith, J., Krasnogor, N., Liò, P., Nicosia, G., Pavone, M. (eds.) *ICARIS 2012*. LNCS, vol. 7597, pp. 14–27. Springer, Heidelberg (2012)
13. Riff, M.C., Montero, E.: A New Algorithm for Reducing Metaheuristic Design Effort. In: *IEEE Congress on Evolutionary Computation (CEC 2013)*, Cancún, México, pp. 3283–3290 (June 2013)

Parameter Prediction Based on Features of Evolved Instances for Ant Colony Optimization and the Traveling Salesperson Problem

Samadhi Nallaperuma, Markus Wagner, and Frank Neumann

Optimisation and Logistics
School of Computer Science
The University of Adelaide, Australia

Abstract. Ant colony optimization performs very well on many hard optimization problems, even though no good worst case guarantee can be given. Understanding the reasons for the performance and the influence of its different parameter settings has become an interesting problem. In this paper, we build a parameter prediction model for the Traveling Salesperson problem based on features of evolved instances. The two considered parameters are the importance of the pheromone values and of the heuristic information. Based on the features of the evolved instances, we successfully predict the best parameter setting for a wide range of instances taken from TSPLIB.

1 Introduction

Ant colony optimization (ACO) [3] has become very popular in recent years to solve a wide range of hard combinatorial optimization problems. Throughout the history of heuristic optimization, attempts have been made to analyze ACO algorithm performance theoretically [6, 17] and experimentally [11, 19]. However, much less work has been done towards the goal of explaining the impact of the problem instance structure and the algorithm parameters on performance.

The study in [19] provides an overview of existing parameter prediction/tuning approaches for ACO in two major directions: (1) parameter choosing before running the algorithm (offline configuration and tuning), and (2) adaptation during runtime (online tuning). It has been shown in [12] that offline tuning outperforms online tuning for the Max-Min Ant System (MMAS) applied to the Traveling Salesperson problem (TSP). However, the drawback of existing offline parameter configuration techniques is that they are time consuming and use a lot of computing power as they need to run iteratively on training instances. We refer the reader to [5] for a discussion on general parameter tuning and prediction methods. To the best of our knowledge, none of the existing approaches have taken structural features of evolved problem instances into consideration when setting the algorithm's parameters.

In early research, the problem hardness analysis of the TSP was based on only a few features that describe the edge cost distribution [14, 20], and the algorithms were typically run on predetermined instances. Later on, more sophisticated methods were

introduced for the instance generation, and the investigated problem features have become more diverse [7, 10, 15]. However, a comprehensive analysis of ACO and TSP problem features has not been conducted so far.

We study the potential of feature-based characterization to be used in automatic algorithm configuration for ACO and consider the well-known Max-Min Ant System [18] for the TSP. One important question in the configuration of ACO algorithms is to what extent the pheromone values and the heuristic information should influence the behaviour of the algorithm—the importance of these two components is determined by the parameters α (for pheromone values) and β (for heuristic information). We first investigate statistical features of evolved (hard, easy, and in-between) instances from [9] and their impact on the appropriate choice of these two parameters. Based on this we build a prediction model in order to predict the right choice for instances of TSPLIB [13]. The potential strength of the prediction model relies on the wide range and on the diversity of the evolved instances, and on the expressiveness of the selected structural features. Our experimental investigations show that the considered features and evolved instances are well suited to predict an appropriate choice for setting the parameters α and β of MMAS.

The outline of the paper is as follows. In Section 2, we introduce the algorithm and the framework of our investigations. In Section 3, we report on easy and hard instances for different parameter combinations and carry out a feature-based analysis. Subsequently, we use these insights to predict parameters for given instances from TSPLIB in Section 4, and we finish with some concluding remarks.

2 Preliminaries

The Traveling Salesperson problem (TSP) is one of the most famous NP-hard combinatorial optimization problems. Given a set of n cities $\{1, \dots, n\}$ and a distance matrix $d = (d_{ij})$, $1 \leq i, j \leq n$, the goal is to compute a tour of minimal length that visits each city exactly once and returns to the origin. We consider the still NP-hard Euclidean TSP, where cities are given by points in the plane and distances are given by the Euclidean distances between these points.

As above-mentioned, our study is focused on the well-known ACO algorithm called Max-Min Ant System (MMAS) [18]. Solutions are constructed by ants visiting cities sequentially, according to a probabilistic formula defined as

$$p_{ij} = \frac{[\tau_{ij}]^\alpha * [\eta_{ij}]^\beta}{\left(\sum_{h \in N_k} [\tau_{ih}]^\alpha * [\eta_{ih}]^\beta\right)},$$

where N_k represents the set of unvisited nodes of ant k , $[\tau_{ih}]$ and $[\eta_{ih}]$ having exponents α and β that represent pheromone and heuristic information respectively. A detailed description and analysis of this algorithm on TSP can be found in the textbook of Dorigo and Stützle (Chapter 3) [3].

To evolve easy and hard instances for the ant algorithms we use the evolutionary algorithm approach previously studied on 2-opt [7] and approximation algorithms [10] for the TSP. The only difference in the instance generation process here is that we

consider several algorithm instances with different parameter settings instead of a single algorithm.

The approximation ratio $\alpha_A(I)$ of an algorithm A for a given instance I is defined as

$$\alpha(I) = A(I)/OPT(I)$$

where $A(I)$ is the tour length produced by algorithm A for the given instance I , and $OPT(I)$ is the value of an optimal solution of I . $OPT(I)$ is obtained by using the exact TSP solver Concorde [2].

3 Features of Hard and Easy Instances

For each ACO algorithm instance with a specific parameter setting of α and β , a set of 100 random TSP instances is generated in two-dimensional unit square $[0, 1]^2$ and placed on a discretized grid. The evolutionary algorithm runs on them for 5000 generations in order to generate a set of hard and a set of easy instances. Each ACO execution is limited to two seconds. In each iteration, the ACO algorithm is run once on a single instance, and then the approximation ratio is calculated. In separate runs, either a higher approximation ratio is favoured to generate hard instances, or a lower ratio is favoured to generate easy instances. This process is repeated for instances of sizes 25, 50, 100 and 200 with the goal of generating easy and hard instances respectively. The instance generation is performed on an Unix cluster with 48 nodes where each node has 48 cores (4 AMD 6238 12-core 2.6Ghz CPUs) and 128GB memory (2.7GB per core).

This study considers 47 features including distances of edge cost distribution, angles between neighbors, nearest neighbor statistics, mode, cluster and centroid features as well as features representing minimum spanning tree heuristics and of the convex hull. A detailed description of these features can be found in [10].

The algorithm parameters considered in this study are the most popular and critical ones in any ACO algorithm, namely the exponents α and β , which represent the influence of the pheromone trails and of the heuristic information respectively. We consider three parameter settings for our analysis: setting 1 represents default parameters ($\alpha = 1$, $\beta = 2$), and settings 2 and 3 represent extreme settings with highest and lowest values in a reasonable range ($\alpha = 0$, $\beta = 4$ and $\alpha = 4$, $\beta = 0$). The general idea behind the choice is that we have to isolate the conditions to investigate the effect which is usually considered in traditional scientific experiments. The rest of the parameters are set in their default values ($\rho = 0.2$, ants = 20) as in the original MMAS implementation by Stützle [16].

3.1 Feature Analysis

Our experimental results for the MMAS with the first parameter setting ($\alpha = 1$, $\beta = 2$) show the following.¹ For the first and the second parameter settings, the standard deviation of angles of the easy instances are significantly smaller than the values of the

¹ Due to space limitations here we present only a few significant findings. We refer to Nallaperuma et al. [9] for some preliminary results of the feature analysis.

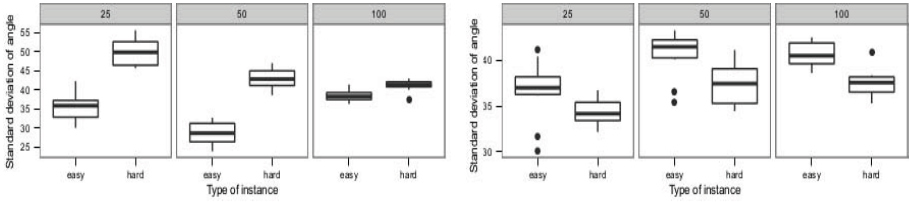


Fig. 1. Boxplots of the standard deviations of the angles between adjacent cities on the optimal tour for parameter setting 2 ($\alpha = 0, \beta = 4$) on the left and setting 3 ($\alpha = 4, \beta = 0$) on the right

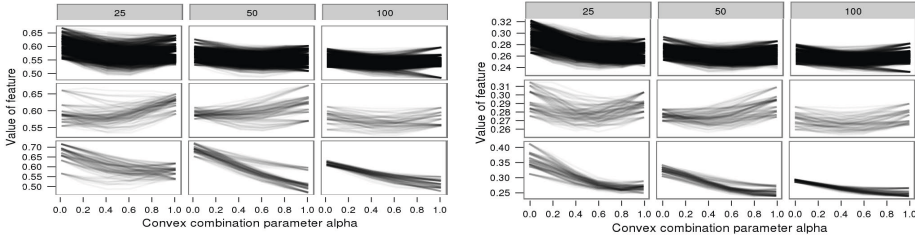


Fig. 2. Feature variation with instance difficulty for mean (left) and standard deviation (right) of distances for the three parameter settings 1 (top), 2 (middle) and 3 (bottom)

hard instances. With increasing instance size, these values change differently for easy and hard instances. Interestingly, this structural difference is even obvious to a human observers who perceive different “shapes” for easy/hard and smaller/larger instances. In contrast to the patterns of the first two parameter settings, the third combination ($\alpha = 4, \beta = 0$) shows an increasing pattern of standard deviation values (with increasing instance size), whereas these values follow a decreasing pattern in the case of the second setting (see Figure 1).

We also study the feature variation for the instances of intermediate difficulty. In order to do this, it is required to generate instances with varying difficulty levels in-between the two extreme difficulties hard and easy. This can be achieved through morphing, where we create instances with varying difficulty levels by forming convex combinations of easy and hard instances. Here, the point matching is done using a greedy strategy where the points of minimum Euclidian distance are matched. These matched instances are then used to produce a set of instances with intermediate difficulty by taking the convex combination based on the convex combination parameter $\alpha_c \in \{0, 0.2, \dots, 0.8, 1\}$ where 0 represents hardest instances and 1 easiest.

Generally, for all three considered parameter settings, most features show similar patterns exhibiting systematic nonlinear relationships with instance difficulty. However, there are a few “contrast patterns” (the feature is increasing in value over instance difficulty for one parameter setting and decreasing for another parameter setting) observed among different parameter settings. For example, the distance mean and the standard deviation show contrast patterns for the second parameter setting ($\alpha = 0, \beta = 4$) from the other two (see Figure 2). Moreover, we observe that the sharp increasing pattern over instance difficulty for the third parameter setting ($\alpha = 4, \beta = 0$) has slowed down

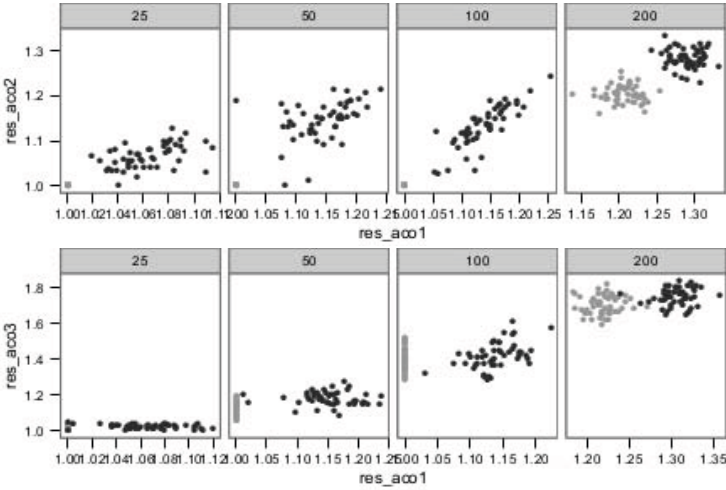


Fig. 3. Performance of the second parameter setting (top) and the third (bottom) on the easy (grey) and hard (black) instances of the first parameter setting

for the default parameter setting, and even converted to a decreasing pattern for the second setting. This provides strong evidence on the impact of parameters. Similar contrast patterns are observed in the other feature groups as well, such as convex hull and nearest neighbour. These contrast patterns suggest the dependence of problem hardness on the algorithm parameters. This dependence further indicates that algorithms with different settings can have complimentary problem-solving capabilities. We believe that such capabilities can provide insights to automatic parameter configuration. Therefore, we further investigate these capabilities by comparing the approximation ratios of the three algorithms achieved on each others' easy and hard instances.

3.2 Comparison of Parameter Settings

As shown in Figure 3, both the second ($\alpha = 0, \beta = 4$) and the third ($\alpha = 4, \beta = 0$) parameter settings have obtained worse approximation ratios for the easy instances of the first parameter setting ($\alpha = 1, \beta = 2$) than the first parameter setting. In the case of the hard instances, the second parameter setting has achieved better approximation ratios than the first parameter setting itself. The outcomes of the other two cross-checks are comparable: given the hard instances of one algorithm configuration, the other two settings achieve better results. This is strong support for our previous conjecture on the complimentary capabilities of different parameter settings.

4 Parameter Prediction

In order to build a reliable model, we significantly extend our collection of data gained from the experiments in Section 3. We generate 1500 instances: 10 hard and easy ones, with sizes 25, 50 and 100, and for each of the 25 parameter combinations $\alpha, \beta \in \{0, 1, 2, 3, 4\}$.

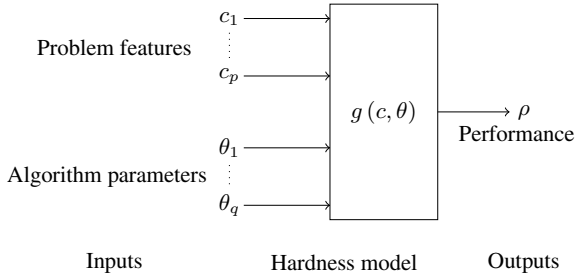


Fig. 4. Prediction Model. It predicts the algorithm performance based on the problem features c_i , $1 \leq i \leq p$ and the possible algorithm parameters θ_j , $1 \leq j \leq q$.

4.1 Prediction Model

We build a simple prediction model merely as a proof of concept that a problem hardness model can be used for ACO parameter prediction. Therefore, we use a popular basic technique for model building. A high level overview of the model is shown in Figure 4. Instead of predicting the allegedly optimal parameters, we actually predict the approximation values given the 25 possible parameter combinations. Then, we select amongst those 25 the combination that achieves the best approximation as the model's output. Hence, the actual model construction is based on the approximation ratio as the dependent variable. Note that a similar model architecture is used in the recent work of Munoz et al. [8] for the prediction of algorithm performance based on landscape features and parameters.

To build our prediction model, we use the classical pattern classification technique introduced by Aha et al. [1], as implemented in the Weka data mining framework [4]. In the training phase, we feed the generated instances into this nearest neighbour search based classifier. As we have seen in the previous hardness analysis, not all problem features appear to be significantly different for easy and hard instances. Consequently, a smaller subset with 15 *strong* features out of the whole (47) feature set is selected: angle mean, angle median, angle sd, centroid mean distance to centroid, centroid max distance to centroid, points on hull, distance mean, distance median, distance max, distance sd, mst distance mean, mst distance max, mst distance sd, nearest neighbour distance sd and nearest neighbour distance coefficient of variance.

4.2 Prediction Results

First, we test our model on a set of 30 randomly generated TSP instances of instance size 100. In the first step, their approximation values are calculated (by averaging the outcomes of 50 repetitions) for all (25) considered parameter combinations. Then the actual best-performing parameter setting is found based on those 25 approximation values (see Table 1 (a) for results).

For more than half of the 30 instances, the model predicts the *correct* minimal approximation ratio, as both winning parameters are the best actual values. Almost all remaining predictions are close to the optimal combination, predicting the actual second best parameter combination. Even though our model is relatively simple, we believe

Table 1. (a): Predicted and actual parameter settings (α , β) for 30 random test instances of size 100. The columns "best" and "second" show the parameter combinations for which the best approximation and second best approximations are achieved in both prediction and actual experiments. **(b):** Results of the Wilcoxon signed rank tests on the predicted and actual approximation ratios for same instances for the hypothesis "actual $>$ predicted" (Test 1) and "predicted $>$ actual" (Test 2), positive rank sum (W) and confidence (p) values are displayed accordingly.

(a)					(b)				
inst	predicted	actual	match	second match	inst	Test 1		Test 2	
	best/second	best/second				W	p-value	W	p-value
1	(4, 4) / (1, 4)	(1, 4)	no	yes	1	99	0.9305	201	0.0714
2	(1, 2)	(1, 3) / (1, 2)	no	yes	2	565.5	0.6797	659.5	0.3221
3	(4, 3)	(4, 3)	yes		3	1296.5	0.5371	1331.5	0.4639
4	(1, 3)	(1, 3)	yes		4	2191	0.6233	2369	0.3774
5	(1, 4)	(1, 4)	yes		5	3404	0.6666	3736	0.3339
6	(1, 4)	(1, 4)	yes		6	4956.5	0.7049	5483.5	0.2954
7	(1, 4)	(1, 4)	yes		7	6808.5	0.7276	7556.5	0.2727
8	(1, 4)	(1, 4)	yes		8	8735	0.8028	9986	0.1973
9	(1, 3)	(1, 3)	yes		9	11117	0.7959	12536	0.2042
10	(2, 3)	(2, 3)	yes		10	14274.5	0.5590	14405.5	0.4411
11	(1, 3)	(1, 3)	yes		11	16863	0.6171	17328	0.3831
12	(1, 3)	(1, 4)/(1, 3)	no	yes	12	20273.5	0.6086	20767.5	0.3915
13	(1, 3)	(1, 3)	yes		13	24308.5	0.4869	23896.5	0.5132
14	(4, 4)	(1, 4)/(4, 4)	no	yes	14	28605	0.4325	27675	0.5676
15	(1, 2)	(1, 4)/(1, 2)	no	yes	15	32799.5	0.4094	31461.5	0.5907
16	(3, 4)	(3, 4)	yes		16	37968.5	0.3269	35567.5	0.6732
17	(1, 4)	(1, 3)/(1, 4)	no	yes	17	43214	0.2709	39814	0.7291
18	(1, 2)	(1, 3)/(1, 2)	no	yes	18	49679	0.1532	43849	0.8468
19	(4, 4)	(2, 4)/(4, 4)	no	yes	19	54671	0.1863	49069	0.8138
20	(1, 3)	(1, 2)/(1, 3)	no	yes	20	61524	0.1248	53916	0.8753
21	(4, 4)	(4, 4)	yes		21	68491.5	0.0712	58264.5	0.9288
22	(1, 4)	(1, 4)	yes	yes	22	75075	0.0710	64053	0.9290
23	(1, 1)	(1, 3)	no	no	23	80356.5	0.1497	71719.5	0.8504
24	(1, 1)	(1, 1)	yes		24	88700.5	0.0856	76899.5	0.9144
25	(1, 4)	(1, 3)/(1, 4)	no	yes	25	96134	0.0750	82967	0.9250
26	(1, 4)	(1, 4)	yes		26	104622.5	0.0627	89753.5	0.9373
27	(1, 2)	(1, 2)	yes		27	113339	0.0552	96937	0.9449
28	(1, 3)	(1, 3)	yes		28	122681.5	0.0362	103446.5	0.9639
29	(1, 3)	(1, 3)	yes		29	130368	0.0564	112188	0.9436
30	(3, 4) / (1, 4)	(1, 4)	no	yes	30	140646.5	0.0394	119634.5	0.9606

that this first result already supports our initial claim that parameters can be predicted based on preceding instance analyses.

Although the model cannot produce the best parameter setting for all instances, the raw approximation values for the predicted and the actual performance are very similar. Therefore, we conduct a rank test to observe any significant difference between the predicted values. We choose the Wilcoxon signed rank test [21], as there is no guarantee about the distribution, and the results are paired as they are based on the TSP instance on which the approximation ratio is obtained. For each TSP instance the predicted and actual approximation ratios obtained for all parameter settings are considered for the test. For the first test we set the hypothesis that the actual values are greater than the predicted values, and then the test is repeated with the counter hypothesis. For both tests and for most instances, the resulting p values are reasonably large, hence both of the alternative hypothesis are rejected (see Table 1 (b)). Thus, we fail to reject the null hypothesis, meaning that both distributions are equal. Only for two instances we

Table 2. Predicted and actual parameter settings (α , β) for 25 TSPLIB instances of size in range 51–264. Note, that the underlying model is based only on our analysis of instances of size 100.

inst	predicted	actual	match	second match
	best/second	best/second		
bier127.tsp	(2, 3)	(2, 3)	yes	
ch150.tsp	(1, 3)/(2,3)	(2, 3)	no	yes
eil51.tsp	(1, 3)	(1, 3)	yes	
kroA100.tsp	(3,3)/(1,3)	(1, 3)	no	yes
kroB100.tsp	(1,3)	(1, 3)	yes	
kroC100.tsp	(1,3)	(1, 3)	yes	
kroD100.tsp	(1,2)	(1, 2)	yes	
pr107.tsp	(2,4)	(2, 4)	yes	
pr76.tsp	(2, 3)	(2, 3)	yes	
st70.tsp	(1, 1)	(1, 1)	yes	
ch130.tsp	(2, 3)	(2, 4)/(2,3)	no	yes
eil101.tsp	(2, 2)/(2,3)	(2, 3)	no	yes
kroA150.tsp	(2, 2)	(2, 2)	yes	
lin105.tsp	(3,2)/(1,3)	(1, 3)	no	yes
pr124.tsp	(4,3)/(1,3)	(1, 3)	no	yes
rat99.tsp	(2, 4)/(1,2)	(1, 4)/(1,3)	no	no
kroB150.tsp	(1, 2)/(1, 3)	(2, 3)/(3,4)	no	no
eli79.tsp	(3,4)/(1,2)	(1, 3)/(1,2)	no	yes
kroE100.tsp	(1,1)/(1,3)	(1, 4)/(1,3)	no	yes
kroA200.tsp	(3, 3)	(3, 3)	yes	
kroB200.tsp	(3, 4)	(3, 4)	yes	
tsp225.tsp	(2, 3)/(3, 4)	(4, 4)/(3, 4)	no	yes
pr264.tsp	(4, 4)	(4, 4)	yes	
gil262.tsp	(2, 3)/(1, 4)	(4, 4)/(4, 3)	no	no
pr226.tsp	(2, 3)/(1, 3)	(4, 3)/(3, 3)	no	no

observe p values less than 0.05, and thus we fail to reject the alternative hypothesis with 95% significance (thus reject the null hypothesis) that they are different.

Second, we test our model on a set of famous benchmark instances from TSPLIB [13] and the results are shown in Table 2. Interestingly, they are qualitatively similar to the results of the test on random TSP instances (Table 1), even though these “real world” instances have never been part of the model building process. Therefore, this second investigation provides further evidence on the accuracy of the model-based performance predictions. We conjecture that the reasons for this strong performance are (1) the large distribution of the training set varying from extreme hard to extreme easy TSP instances and (2) the strength of the selected feature set in expressing problem hardness for ACO algorithm instances with specified parameter settings. In order to use this model for prediction, a very short preprocessing step is required that calculates the 15 above-mentioned feature values for the input instance.

5 Conclusions

In this paper, we have shown how to predict the parameter setting of ACO algorithms based on features of evolved problem instances. We considered the parameters α and β which determine the importance of the pheromone concentration and heuristic information, respectively. Based on instance features for the classical Traveling Salesperson Problem, we built a prediction model to determine the values of α and β . Our investigations on a wide range of instances from TSPLIB show that the instance features allow

for a reliable prediction of well-performing algorithm setups. For future work, we plan on improving the prediction model by integrating other ACO parameters such as the number of ants and the pheromone update strength.

Acknowledgements. We thank Bernd Bischl for early discussions, Heike Trautmann and Olaf Mersmann for the feedback on the preliminary version of this research. This research has been supported by the Australian Research Council (ARC) under grant agreement DP140103400.

Bibliography

- [1] Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. *Mach. Learn.* 6(1), 37–66 (1991)
- [2] Applegate, D., Cook, W.J., Dash, S., Rohe, A.: Solution of a Min-Max Vehicle Routing Problem. *Journal on Computing* 14(2), 132–143 (2002)
- [3] Dorigo, M., Stützle, T.: *Ant Colony Optimization*. Bradford Company (2004)
- [4] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. *SIGKDD Explorations Newsletter* 11(1), 10–18 (2009)
- [5] Hoos, H.: Automated algorithm configuration and parameter tuning. In: Hamadi, Y., Monfroy, E., Saubion, F. (eds.) *Autonomous Search*, pp. 37–71. Springer, Heidelberg (2012)
- [6] Kötzing, T., Neumann, F., Röglin, H., Witt, C.: Theoretical analysis of two ACO approaches for the traveling salesman problem. *Swarm Intelligence* 6, 1–21 (2012)
- [7] Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., Neumann, F.: A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. In: *Annals of Mathematics and Artificial Intelligence*, pp. 1–32 (2013)
- [8] Muñoz, M.A., Kirley, M., Halgamuge, S.K.: A meta-learning prediction model of algorithm performance for continuous optimization problems. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I. LNCS*, vol. 7491, pp. 226–235. Springer, Heidelberg (2012)
- [9] Nallaperuma, S., Wagner, M., Neumann, F.: Ant colony optimisation and the traveling salesperson problem: Hardness, features and parameter settings (extended abstract). In: *15th Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion (GECCO Companion)*, pp. 13–14. ACM (2013)
- [10] Nallaperuma, S., Wagner, M., Neumann, F., Bischl, B., Mersmann, O., Trautmann, H.: A Feature-based Comparison of Local Search and the Christofides Algorithm for the Traveling Salesperson Problem. In: *International Conference on Foundations of Genetic Algorithms, FOGA* (2013)
- [11] Pellegrini, P., Favaretto, D., Moretti, E.: On $\mathcal{MA}\mathcal{X}$ – \mathcal{MZN} ant system’s parameters. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) *ANTS 2006. LNCS*, vol. 4150, pp. 203–214. Springer, Heidelberg (2006)
- [12] Pellegrini, P., Stützle, T., Birattari, M.: Off-line vs. on-line tuning: A study on $\mathcal{MA}\mathcal{X}$ – \mathcal{MZN} ant system for the TSP. In: Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., Stützle, T. (eds.) *ANTS 2010. LNCS*, vol. 6234, pp. 239–250. Springer, Heidelberg (2010)
- [13] Reinelt, G.: TSPLIB – A Traveling Salesman Problem Library. *ORSA Journal on Computing* 3(4), 376–384 (1991)

- [14] Ridge, E., Kudenko, D.: Determining Whether a Problem Characteristic Affects Heuristic Performance. In: Cotta, C., van Hemert, J. (eds.) *Recent Advances in Evol. Comp. SCI*, vol. 153, pp. 21–35. Springer, Heidelberg (2008)
- [15] Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Blum, C., Battiti, R. (eds.) *LION 4. LNCS*, vol. 6073, pp. 266–280. Springer, Heidelberg (2010)
- [16] Stützle, T.: Software package: Acotsp.v1.03.tgz (2012)
- [17] Stützle, T., Dorigo, M.: A short convergence proof for a class of Ant Colony Optimization algorithms. *IEEE Trans. on Evolutionary Computation*, 358–365 (2002)
- [18] Stützle, T., Hoos, H.H.: MAX-MIN Ant system. *Future Generation Computer Systems* 16(9), 889–914 (2000)
- [19] Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., Montes de Oca, M., Birattari, M., Dorigo, M.: Parameter Adaptation in Ant Colony Optimization. In: *Autonomous Search*, pp. 191–215. Springer (2012)
- [20] Stützle, T., Hoos, H., Merz, P.: An Analysis of the Hardness of TSP Instances for Two High-performance Algorithms. In: *6th Metaheuristics International Conference (MIC)*, pp. 361–367 (2005)
- [21] Wilcoxon, F.: Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1(6), 80–83 (1945)

Self-Adaptive Genotype-Phenotype Maps: Neural Networks as a Meta-Representation

Luís F. Simões¹, Dario Izzo², Evert Haasdijk¹, and Agoston Endre Eiben¹

¹ Vrije Universiteit Amsterdam, The Netherlands

² European Space Agency, The Netherlands

{luis.simoes,e.haasdijk,a.e.eiben}@vu.nl, dario.izzo@esa.int

Abstract. In this work we investigate the usage of feedforward neural networks for defining the genotype-phenotype maps of arbitrary continuous optimization problems. A study is carried out over the neural network parameters space, aimed at understanding their impact on the locality and redundancy of representations thus defined. Driving such an approach is the goal of placing problems' genetic representations under automated adaptation. We therefore conclude with a proof-of-concept, showing genotype-phenotype maps being successfully self-adapted, concurrently with the evolution of solutions for hard real-world problems.

Keywords: Genotype-Phenotype map, Neuroevolution, Self-adaptation, Adaptive representations, Redundant representations.

1 Introduction

Automated design of Evolutionary Algorithms (EAs) has been on the research agenda of the Evolutionary Computing community for quite some years by now. The two major approaches for finding good values for the numeric and/or symbolic parameters (a.k.a. EA components) are parameter tuning and parameter control, employed before or during the run, respectively. The current state of the art features several techniques to find 'optimal' values or instances, for all parameters and components, with one notable exception: the genotype-phenotype (G-P) mapping, a.k.a. the representation. In contrast to all other parameters related to selection, variation, and population management, there are only a handful of papers devoted to adapting representations. Considering the widely acknowledged importance of having a good representation, the lack of available techniques to 'optimise' it is striking. One could say that the challenge of tuning and/or controlling the representation in an EA is the final frontier in automated EA design.

In this paper we investigate the possibility of using adjustable genotype-phenotype maps for continuous search spaces. In particular, we propose neural networks (NN) as the generic framework to represent representations (i.e., NNs as a meta-representation). This means that for a given phenotype space $\Phi_p \subset \mathbb{R}^n$ and a genotype space $\Phi_g \subset \mathbb{R}^m$ the set of all possible representations we consider is the set of all NNs mapping Φ_g to Φ_p .

In the technical sections of the paper we investigate this approach, with regard to its expressiveness, and learnability. Informally, we consider a meta-representation expressive if it is versatile enough to define a wide range of transformations, granting it the potential power to restructure arbitrary fitness landscapes into more efficiently explorable genotype spaces for the underlying optimizer. As for learnability, we are pragmatic. All we want to demonstrate at this stage, is the existence of a learning mechanism that can change the NN during the run of the given EA, in such a way that the EA performance (measured by solution quality) is improved. Thus, in terms of the classic tuning-control division, we want to demonstrate the existence of a good control mechanism for NN-based representations.

2 Related Work

A fundamental theoretical result by Liepins & Vose [7], shows that “virtually all optimizable (by any method) real valued functions defined on a finite domain [are] theoretically easy for genetic algorithms given appropriately chosen representations”. However, “the transformations required to induce favorable representations are generally arbitrary permutations, and the space of permutations is so large that search for good ones is intractable”. Nevertheless, [7] still calls for research into approaches that adapt representations at a meta-level. In particular, [7] shows affine linear maps to provide sufficient representational power to transform fully deceptive problems into easy ones.

In the over two decades since [7], a vast body of work emerged, addressing ways to automatically adapt and control all sorts of Evolutionary Algorithm components [4,8]. The genetic representation, and its genotype-phenotype map, however, despite their recognized role as critical system components, have only sporadically been addressed in the literature. De Jong, in [2], considers that “perhaps the most difficult and least understood area of EA design is that of adapting its internal representation.”

In a series of papers culminating in [3], Ebner, Shackleton & Shipman proposed several highly-redundant G-P maps, the cellular automaton (CA) and random boolean network (RBN) mappings, in particular, being of special relevance to the present research. In them, chromosomes are composed of a dynamical system’s definition (CA or RBN rule table, and also the cell connectivity graph in the case of RBN), along with the system’s initial state. Decoding into the phenotype space takes place by iterating the dynamical system for a number of steps, from its initial state, according to the encoded rule table.

3 Neural Networks as Genotype-Phenotype Maps

We have chosen to use neural networks as the basis of our approach for two reasons. First, they are global function approximators. In principle, they are capable of expressing any possible G-P map (given a sufficiently large number of

hidden layer neurons). Second, they are learnable, even evolvable. There is much experience and know-how about ‘optimising’ neural nets by evolution [5,11].

Formally, and following Rothlauf’s notation [9, Sec. 2.1.2], we then have that, under an indirect representation scheme, evolution is guided by a fitness function f that is decomposed into a genotype-phenotype map f_g , and a phenotype-fitness mapping f_p . Genetic operators such as recombination and mutation are applied over genotypes $x^g \in \Phi_g$ (where Φ_g stands for the genotypic search space). A genotype-phenotype map $f_g : \Phi_g \rightarrow \Phi_p$ decodes genotypes x^g into their respective phenotypes $x^p \in \Phi_p$ (where Φ_p is the phenotypic space), and fitness assignment takes place through $f_p : \Phi_p \rightarrow \mathbb{R}$, which maps phenotypes into fitness values. In summary, individuals in the population of genotypes are evaluated through $f = f_p \circ f_g$, the fitness of a genotype x^g being given by $f(x^g) = f_p(f_g(x^g))$. When considering an EA that searches in a continuous genotypic space, for solutions that decode into continuous phenotypes, we then have that $\Phi_g \subset \mathbb{R}^m$, and $\Phi_p \subset \mathbb{R}^n$, where m and n stand, respectively, for the dimensionalities of the considered genotypic and phenotypic spaces. A genotype-phenotype map is then a transformation $f_g : \mathbb{R}^m \rightarrow \mathbb{R}^n$.

Let \mathcal{N} be a fully connected, feedforward neural network with l layers and d^k neurons on its k -th layer ($k = 1..l$). If \mathcal{L}^k is the vector representing the states of the d^k neurons in its k -th layer, then the network’s output can be determined through

$$\mathcal{L}_i^k = \sigma(b_i^k + \sum_{j=1}^{d^{k-1}} w_{ij}^k \mathcal{L}_j^{k-1}), i = 1..d^k,$$

where b^k represents the biases for neurons in the k -th layer, and w_i^k the weights given to signals neuron \mathcal{L}_i^k gets from neurons in the preceding layer. A sigmoidal activation function $\sigma(y) = 1/(1 + e^{-y})$ is used throughout this paper. The network’s output, \mathcal{L}^l , is then uniquely determined through b , w , and \mathcal{L}^1 , the input vector fed to its input layer.

Without loss of generality, in the sequel we assume that the given phenotype space is an n dimensional hypercube. (If needed, the interval $[0, 1]$ can be mapped with a trivial linear transformation to the actual user specified lower and upper bounds for each variable under optimization.) Using a neural network as a G-P map, we then obtain a setup where the number of output neurons $d^l = n$ and the mapping itself is $f_g : [0, 1]^{d^1} \rightarrow [0, 1]^{d^l}$. To specify a given G-P mapping network we will use the notation $\mathcal{N}(m_p, m_a)$, where m_p and m_a are the map parameters¹ and map arguments, defined as follows. The vector $m_p \in [-1, 1]^d$ contains the definition of all weights and biases in the network, while the vector m_a designates the input vector fed into the network. With this notation, we obtain a formal framework where genotypes are map arguments to the neural net and the representation is $f_g = \mathcal{N}(m_p, \cdot)$. Given a genotype $x^g \in [0, 1]^{d^1}$, the corresponding phenotype is $f_g(x^g) = \mathcal{N}(m_p, x^g) \in [0, 1]^{d^l}$.

As shorthand for a considered NN architecture, we will use notation such as 30-5-10, to indicate a fully connected feedforward neural network with $l = 3$

¹ “Map parameters” named by analogy with the strategy parameters (e.g., standard deviations of a Gaussian mutation) traditionally used in Evolution Strategies.

layers, having $d^1 = 30$ neurons in its input layer, $d^2 = 5$ neurons in the hidden layer, and $d^3 = 10$ neurons in the output layer ($\Phi_g = [0, 1]^{30}$, $\Phi_p \subset \mathbb{R}^{10}$).

4 Expressiveness

The expressiveness challenge facing a representation of G-P maps, is that of ensuring the “language” used to represent representations supports the specification of widely distinct transformations between both spaces. Given our use of neural networks to represent G-P maps, and the knowledge that their expressiveness needs to be traded-off with their learnability, we will then address the following research question:

- what is the expressiveness retained by small to medium sized neural networks?

We will focus our analysis on two often-studied representation properties: *locality* and *redundancy* [9,1]. A representation’s locality [9, Sec. 3.3] describes how well neighboring genotypes correspond to neighboring phenotypes. In a representation with perfect (high) locality, all neighboring genotypes correspond to neighboring phenotypes. Theoretical and experimental evidence [9,1] support the view that high locality representations are important for efficient evolutionary search, as they do not modify the complexity of the problems they are used for. A redundant encoding, as the name implies, provides multiple ways for a phenotype to be encoded in the genotype. In [9, Sec. 3.1] different kinds of redundancies are identified, the advantages and disadvantages of each one being then subjected to theoretical and experimental study.

4.1 Map Characterization

We characterize here the expressive power of different NN architectures, in terms of the locality and redundancy of the G-P maps they can define. We conduct our analysis over the G-P map design space, by sampling random NN configurations within given architectures.

Setup. The G-P map design space is explored by randomly sampling (with uniform probability) NN weights and biases, in the range $[-1, 1]$, thus providing the definition of map parameters, m_p . We follow by generating a large number of map arguments, m_a (10000, to be precise), in the range $[0, 1]$, according to a quasi-random distribution. Sobol sequences are used to sample the genotype space ($m_a \in \Phi_g$), so as to obtain a more evenly spread coverage. The m_a scattered in the genotype space are subsequently mapped into the phenotype space. The Euclidean metric is used to measure distances between points within both the genotype and phenotype spaces.

The analysis conducted here is fitness function independent, but for illustration purposes (Figure 1), and for defining the phenotypic space in which distances will be measured, we consider the well known Rastrigin function².

² Rastrigin: $f_p(x_1^p, \dots, x_n^p) = 10n + \sum_{i=1}^n [(x_i^p)^2 - 10 \cos(2\pi x_i^p)]$, $\Phi_p = [-5.12, 5.12]^n$.

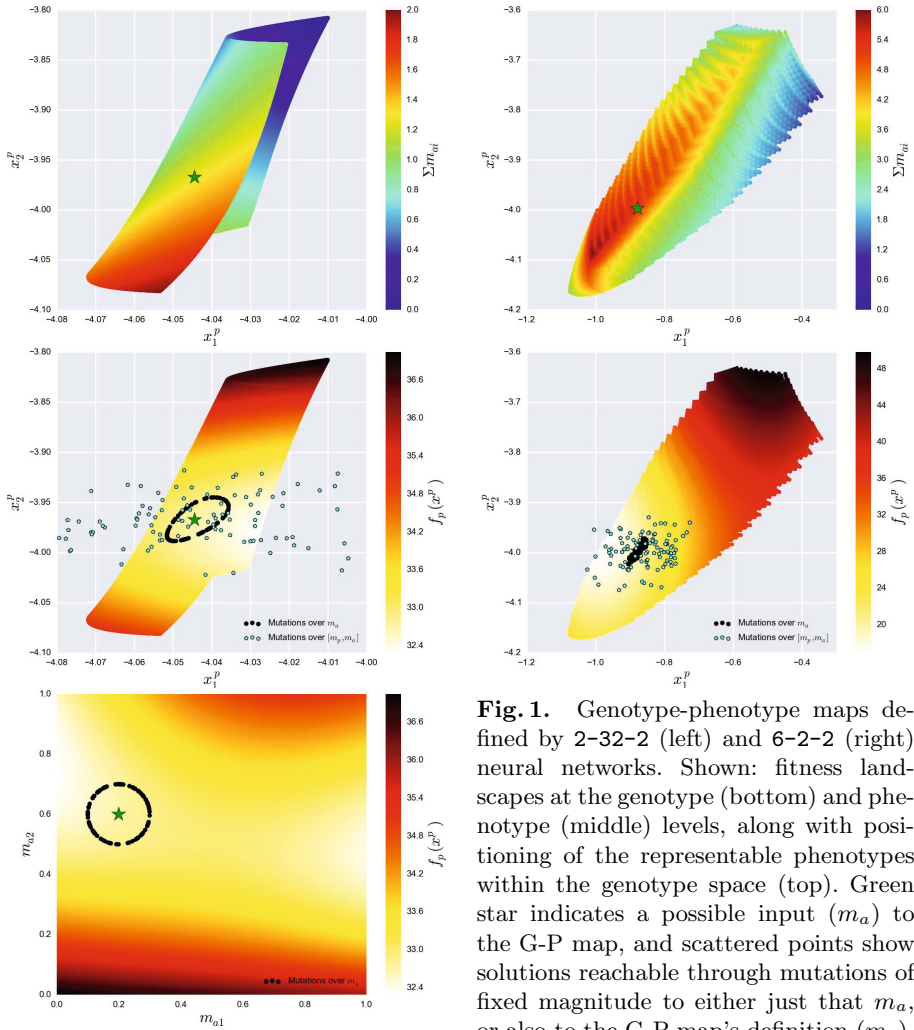


Fig. 1. Genotype-phenotype maps defined by 2-32-2 (left) and 6-2-2 (right) neural networks. Shown: fitness landscapes at the genotype (bottom) and phenotype (middle) levels, along with positioning of the representable phenotypes within the genotype space (top). Green star indicates a possible input (m_a) to the G-P map, and scattered points show solutions reachable through mutations of fixed magnitude to either just that m_a , or also to the G-P map's definition (m_p).

Measuring locality. We characterize the locality of representations definable by a given neural network architecture, by randomly sampling the space of possible network configurations (m_p) in that architecture. A sample of 1000 points is taken, out of the 10000 m_a given by the Sobol sequence (mentioned above), and a mutated version generated. A mutation is always a random point along the surface of the hypersphere centered on the original genotype, and having a radius equal to 1% the maximum possible distance in the Φ_g hypercube. The mutated m_a is mapped into the phenotype space, and its distance there to the original point's phenotype measured. Given we wish to consider phenotype spaces having distinct numbers of dimensions, and importantly, given the fact that each different G-P map encodes a different subset of the phenotype space, it becomes important to normalize phenotypic distances, in a way that makes

them comparable. To that end, we identify the hyperrectangle that encloses all the phenotypes identified in the initial scatter of 10000 points, and use the maximum possible distance value there to normalize phenotype distances.

Measuring redundancy. To characterize a representation’s redundancy, we will want to relate phenotypes to the distinct genotypes capable of encoding them. In a representation with high locality, the extent to which dissimilar genotypes express similar phenotypes, provides an indication of its redundancy.

Given a randomly generated NN, and the set of 10000 solutions sampled in genotype space through a Sobol sequence (as previously described), we randomly select 200 of the obtained phenotypes. For each, we perform a k-Nearest Neighbor search, in phenotype space, for its 5 closest phenotypes. Having all our phenotypes been obtained from known genotypes, we are then able to map those nearest neighbors back to the genotypes that led to their expression. One data point in our analysis is then composed of the distances in both genotype and phenotype spaces, between a queried solution, and one of those neighbors. Analysis of one NN is in this setup then given by $200 \cdot 5$ data points. To allow for comparisons of distances across NN architectures, phenotype distances are normalized over the maximum possible distance within the hyperrectangle enclosing the space of 10000 phenotypes obtained through the Sobol process. Additionally, also distances in genotype space are normalized, in this case over the maximum possible distance within the d^1 -dimensional hypercube.

Results. Figure 1 shows the mappings defined by two randomly defined G-P maps (plots obtained by spreading an evenly spaced grid of 10^6 m_a in genotype space, mapping them to phenotype space, and evaluating them). Top panels show the phenotypes expressible by each G-P map, color coded based on the m_a that led to their expression. The continuous color gradients observed in each case show that nearby genotypes are being decoded into nearby phenotypes, and are therefore evidence of two high locality representations (low locality would have resulted in a randomization of the colors present in small neighborhoods). In Figure 1 (top left) we see a genotype space that folds on itself as it gets mapped onto the phenotype space, leading *some* of its expressible phenotypes to become redundantly representable, as seen in the bottom and middle panels.

Figure 2 characterizes the locality of representations definable by different NN architectures. Each of the shown distributions was obtained by analyzing 1000 randomly generated G-P maps having that architecture, and thus represents a total of 10^6 measured phenotype distances. We consistently observe high locality representations resulting from *all* studied NN architectures: a mutation step of 1% the maximum possible distance in genotype space is in all cases expected to take us across a distance in phenotype space of at most $\sim 1\%$ the maximum possible distance among phenotypes representable by the considered G-P map.

Figure 3 characterizes the redundancy of representations definable by different NN architectures. Each of the shown bivariate distributions was obtained by analyzing 1000 randomly generated G-P maps having that architecture, and thus

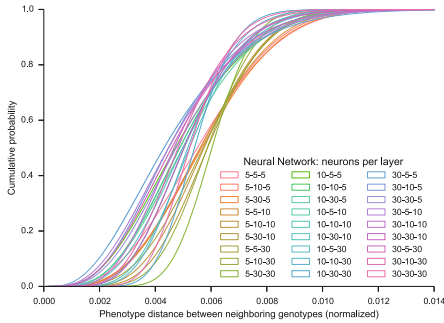


Fig. 2. Locality of representations expressible by different sized neural networks. Shown: empirical cumulative distribution functions of distances in phenotype space between pairs of neighboring genotypes.

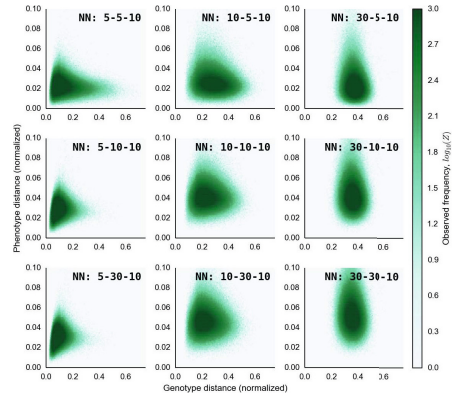


Fig. 3. Redundancy of representations expressible by different sized neural networks. Shown: multivariate distribution relating a phenotype's distance to one of its k -nearest neighbors, with the distance at which the pair lies in genotype space.

represents a total of 10^6 measured phenotype and genotype distances. We clearly see the number of neurons in the input layer as the primary factor determining the expected degree of redundancy in NNs sampled according to a given architecture. Nearest neighbors, which in all cases lie at approximately the same distance in phenotype space (on average, 2 to 6% the span of possible distances in Φ_p), turn out to be representable through genotypes at distances between themselves going on average from roughly 10, to 25, and 40% the span of possible distances in Φ_g , as the number of input neurons grows from 5 to 30.

Discussion. The view emerging from these results is that NN-based G-P maps, as defined in Section 3, tend to generate high-locality representations, with a tuneable degree of expected redundancy.

Given a random G-P map, its genotype space will most likely decode into a limited region of the full phenotype space (as seen in Figure 1). This could be problematic if G-P maps were to be defined off-line, for unknown search spaces, where the risk of being unable to express the global optimum would be considerable. In an on-line adaptation scenario, however, the G-P map is instead at every point of the search devoting its resources to learning a representation of a momentarily relevant portion of phenotype space, while retaining the power to adapt itself, towards expression of newly identified superior phenotypic regions.

5 Learnability

In this section we want to establish the existence of a learning mechanism, that can change the NN-based G-P map during the run of a given EA, in such a way that the EA performance (measured by solution quality) is improved.

When designing a suitable learning mechanism we face a couple of principal decisions, namely: which learning mechanism to use, and how to measure G-P maps' quality, so as to steer the learning mechanism towards superior ones? We answer these questions by employing a self-adaptation scheme, whereby the parameter vector that specifies a G-P map, m_p , is added to the chromosome and co-evolves with the solution (m_a). By the very nature of self-adaptation, this option elegantly solves the second problem. In particular, we do not need to specify an explicit quality measure for the G-P map. Instead, the G-P map is evaluated implicitly: it is good if it leads to good solutions.

In neuroevolution, recombination-based EAs are known to face some difficulties when optimizing NNs. This is known as the competing conventions problem [5,11]. To avoid this problem, we decide to use a mutation-only EA.

Having made these choices, the issue of learnability addressed in this section can now be phrased as follows:

- can a self-adaptive mechanism within a mutation-only EA effectively learn useful G-P mappings that lead to increased EA performance?

5.1 Experimental Evaluation

We experimentally evaluate here the performance achieved by EA setups that self-adapt G-P maps, through comparison with identical optimizer setups that work instead directly over the phenotype space.

Setup. Our experimental validation compares, for the same problem, optimization using a direct representation $x^g = x^p$, against optimization using an indirect representation, where a NN-based G-P map is self-adapted in the chromosome, together with its input, $x^g = [m_p, m_a]$.

We evaluate the different setups by searching for solutions to the Cassini 1 and Messenger_full problems. These are difficult, real-world problems, of spacecraft interplanetary trajectory design. Their full specification can be found online, in the GTOP Database [10]³. In the indirect representation experiments we demonstrate the usage of distinct neural network architectures: in Cassini 1 we ask the optimizer to learn and exploit highly redundant encodings of the phenotype space, by using 20-3-6 G-P maps; In Messenger_full we ask it instead to learn minimally redundant, lower dimensional projections of the problem's 26-dimensional phenotype space, by using 2-2-26 G-P maps.

We make use of the Improved Fast Evolutionary Programming (IFEP) algorithm introduced in [12], extended with the success rate based dynamic lower bound adaptation (DLB1) described in [6]. The optimizer was tuned as follows: population size $\mu = 25$, and tournament size $q = 2$. The strategy parameters in individuals' chromosomes, which parameterize their mutation operators, were initialized to values of $\eta = 0.03$, and had initial lower bounds of $\eta_- = 0.015$, adapted by DLB1, every 5 generations, using a reference success rate of $A = 0.3$.

³ <http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html>

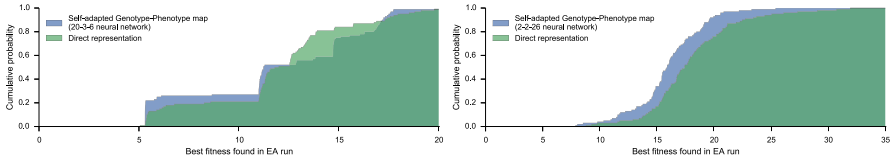


Fig. 4. Comparison of direct and indirect representations on the Cassini 1 (left) and Messenger_full (right) problems. Shown: empirical cumulative distribution functions of the best fitness value reached in an EA run.

In each IFEP run, evolution proceeded for a total of 5000 generations. IFEP has each parent generating two mutated offspring, one according to a Gaussian, and another according to a Cauchy distribution⁴. As such, 50 solutions were generated per generation, leading to a total of 250000 fitness evaluations per run. The indirect representation setups used bounds of $m_{pi} \in [-1, 1]$, and $m_{ai} \in [0, 1]$. For increased fairness in the comparison between direct and indirect representations, chromosomes were in both cases normalized at the optimizer level, into the unit hypercube, $x^g \in [0, 1]^d$, and scaled back at decoding and evaluation time.

Results. Figure 4 presents the distributions of best found solutions’ fitness values, in 100 independent EA runs performed under each optimization setup.

Both Cassini 1 and Messenger_full are minimization problems. In Cassini 1 we see a median fitness of 11.9 being found with a direct representation, and 11.2 with an indirect one. Peak performance was however lower on the indirect representation runs: 5.3, against 5.1 for the direct representation. In Messenger_full the indirect representation improved median performance from 17.4, to 15.9, as well as peak performance (7.9, against 8.8). We see also in it a considerable improvement to worst case performance (from 32.0 to 25.4).

Analysis. Extending chromosomes with the definition of their own G-P maps, naturally places a significant burden on top of the optimization process: Cassini 1 goes from being a 6-dimensional optimization problem in the direct representation case, to a 107-dimensional one when simultaneously learning a 20–3–6 G-P map. Similarly, the Messenger_full problem goes from 26 to 86 dimensions when adding a 2–2–26 G-P map. Still, as seen in Figure 4, EA performance is matched, or even surpassed, by the G-P maps’ addition.

Back in Section 4 we saw in Figure 1 (middle panels), regarding mutation over vectors containing $[m_p, m_a]$, that it is possible to conduct a robust search simultaneously over the G-P map’s definition, and its inputs: mutated offspring tend to encode phenotypes in the vicinity of those of their parents. The results reported in this section show that such variation, in an evolutionary setting,

⁴ The lognormal self-adaptation of strategy parameters employed by EP is not used in the indirect representation setups to adapt the (also self-adapted) map parameters. Instead, their variation takes place through the Gaussian (or Cauchy) mutation.

indeed allows for the self-adaptation of G-P maps to take place concurrently with the search for problem solutions.

6 Conclusion

We investigated the usage of neural networks as a meta-representation, suited to the encoding of genotype-phenotype maps for arbitrary pairings of fitness landscapes and metaheuristics that are to search on them.

Small to moderately sized feedforward neural networks were found to define, on average, high locality representations (where structure of the phenotypic fitness landscape is locally preserved in the genotype space), and having a degree of redundancy tuneable through the number of neurons in the input layer.

An exploration into the feasibility of evolving genotype-phenotype maps, concurrently with the problem solution, showed this to be a viable approach.

Acknowledgements. Luís F. Simões was supported by FCT (Ministério da Ciência e Tecnologia) Fellowship SFRH/BD/84381/2012.

References

1. Correia, M.B.: A study of redundancy and neutrality in evolutionary optimization. *Evolutionary Computation* 21(3), 413–443 (2013)
2. De Jong, K.: Parameter Setting in EAs: a 30 Year Perspective. In: Lobo, et al. (eds.) [8], pp. 1–18
3. Ebner, M., Shackleton, M., Shipman, R.: How neutral networks influence evolvability. *Complexity* 7(2), 19–33 (2001)
4. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 3(2), 124–141 (1999)
5. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. *Evolutionary Intelligence* 1(1), 47–62 (2008)
6. Liang, K.H., Yao, X., Newton, C.S.: Adapting Self-Adaptive Parameters in Evolutionary Algorithms. *Applied Intelligence* 15(3), 171–180 (2001)
7. Liepins, G.E., Vose, M.D.: Representational issues in genetic optimization. *Journal of Experimental & Theoretical Artificial Intelligence* 2(2), 101–115 (1990)
8. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): *Parameter Setting in Evolutionary Algorithms*. SCI, vol. 54. Springer, Heidelberg (2007)
9. Rothlauf, F.: *Representations for Genetic and Evolutionary Algorithms*, 2nd edn. Springer, Heidelberg (2006)
10. Vinkó, T., Izzo, D.: Global optimisation heuristics and test problems for preliminary spacecraft trajectory design. ACT technical report GOHTPPSTD. European Space Agency, the Advanced Concepts Team (September 2008)
11. Yao, X.: Evolving artificial neural networks. *Proceedings of the IEEE* 87(9), 1423–1447 (1999)
12. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming Made Faster. *IEEE Transactions on Evolutionary Computation* 3(2), 82–102 (1999)

The Baldwin Effect Hinders Self-Adaptation

Jim Smith

Department of Computer Science and Creative Technologies,
University of the West of England,
Bristol, BS161QY, UK
james.smith@uwe.ac.uk
<http://www.cems.uwe.ac.uk/~jsmith>

Abstract. The “end-game” of evolutionary optimisation is often largely governed by the efficiency and effectiveness of searching regions of space known to contain high quality solutions. In a traditional EA this role is done via mutation, which creates a tension with its other different role of maintaining diversity. One approach to improving the efficiency of this phase is self-adaptation of the mutation rates. This leaves the fitness landscape unchanged, but adapts the shape of the probability distribution function governing the generation of new solutions. A different approach is the incorporation of local search – so-called Memetic Algorithms. Depending on the paradigm, this approach either changes the fitness landscape (Baldwinian learning) or causes a mapping to a reduced subset of the previous fitness landscape (Lamarckian learning). This paper explores the interaction between these two mechanisms. Initial results suggest that the reduction in landscape gradients brought about by the Baldwin effect can reduce the effectiveness of self-adaptation. In contrast Lamarckian learning appears to enhance the process of self-adaptation, with very different behaviours seen on different problems.

1 Introduction

Evolutionary Algorithms (EAs) are a class of population-based global search heuristics that have proved highly successful in many optimisation domains [5]. Randomised mutation and crossover operators create a non-uniform probability distribution function (pdf) over the search space for sampling new candidate solutions. The shape of this pdf is governed by a parent pool selected from the current population, the choice of recombination and mutation operators, and their associated parameters. A broader pdf allows exploration of the search space, and hence the ability to escape local optima. A narrower pdf allows exploitation of hard-won information by focussing sampling in the vicinity of promising solutions. The way in which the trade-off between these two factors is managed has a major impact on both the effectiveness and efficiency of search.

One common approach is to couple the randomised nature of EAs with a more systematic local search method to create Memetic Algorithms (MAs). This may be done in a number of ways – see e.g. [9] for a description and taxonomy and [11] for a recent survey. This paper will examine the simplest and most common:

after recombination and mutation, each offspring undergoes local search for a specified number of iterations. In a Baldwinian paradigm, akin to life-term learning, the offspring has its fitness replaced with that of the fittest neighbour found by the local search. The Lamarckian paradigm is more drastic – both the “genome” and fitness of the offspring are replaced. Studies of these two paradigms with the EC date back to the mid-1990s. Both process alter the search landscape “seen” by the EA, but in different ways (see Section 2).

Another very common approach, is parameter adaptation. Typically an initially more uniform pdf is “narrowed” to focus more on promising regions of the search space over time. In both the combinatorial and real-valued domains, the majority of research and applications have focussed on adapting the mutation parameters [4]. Whether adaptation is driven implicitly (e.g. via self-adaption) or explicitly via the application or an “external” algorithm, a key factor is the presence of some form of evidence of the utility of an operator, or parameter value in generating high quality solutions from the current population. In the self-adaptive paradigm the evidence is implicit - successful strategies are those that produce offspring that survive, and increase their representation via association with above average quality solutions. These approaches have been successfully combined with a focus on adaptation at the memetic level [18,19], but little or no attention has been paid to the potential issues even with simple “first-generation” MAs, when the action of local search potentially destroys the link between strategies and offspring survival that is considered essential for successful self-adaptation to occur.

This paper represents a start at understanding this issue by examining the patterns of behaviour observed when applying a simple MA with self-adaptation of mutation rates to some well-understood combinatorial problems, where the “building blocks” are of different orders, so that some cannot be discovered by local search alone. Specifically it examines the following hypotheses: **H1** One-step Baldwinian learning has a “blurring” effect on the fitness landscape that reduces the effect of different mutation rates, and hence the selection pressure between them, hindering effective self-adaptation; **H2** One-step Lamarckian learning behaves differently. The mapping to a reduced search space occurring when offspring are replaced by fitter neighbours effectively increases the selection pressure towards lower mutation rates; **H3** On problems with single-bit building blocks, using multiple steps of local search compounds the effects seen above and increases the selective pressure towards lower mutation rates; **H4** In contrast, on problems with higher order building blocks, the effect of multiple steps of local search is to act as a repair function, which preserves higher mutation rates.

Section 2 provides a brief introduction to the key concepts. Section 3 describes the algorithms, test problems, and methods used to generate and analyse results. The results are presented in Section 4 and discussed in Section 5. Section 6 draws conclusions and suggests future work.

2 Background

The practice and theory of self-adaptation of mutation rates has been documented in the continuous domain since the 1960s (see, e.g., [14,3]), the binary domain since the 1990s (see [1,20,16]) and more recently for permutations [15]. A recent survey is [10]. To achieve the necessary selection pressure it has been found preferable to use a survivor:offspring ration of around 1:5 which tallies with previous work in Evolution Strategies. For combinatorial problems there is evidence that the use of a continuous variable to encode for the mutation rate, and subject to log-normal adaptation, can be outperformed by a simpler scheme [21,16,17]. In this scheme the gene encoding for the mutation rate has a discrete set of alleles, and when itself subject to mutation is randomly reset with a small probability. In particular it was shown that the mechanism for adapting the encoded mutation rate is important – allowing the operator to work “on-itself” (as per [2,20]) will lead to premature convergence to sub-optimal attractors. Similar theoretical [13] and experimental (e.g. [7]) results have been found in the continuous domain. Extensive experimentation revealed that in binary search spaces different variants of self-adaptation do offer performance advantages [12], but that a deeper understanding of the processes involved is still needed.

The field of Memetic Computation encompasses a wide range of algorithms based on the concept of memes as methods for generating or improving individual solutions to one or more problem instances. Ong et. al. [11] consider a more general paradigm which uses “*the notion of meme(s) as units of information encoded in computational representations for the purposes of problem solving*”. This enticing view nevertheless requires a better understanding of the basic processes at work before more complex systems can be built. Therefore this paper is restricted to a simple first generational memetic algorithm where a greedy local search mechanism is applied to each offspring after it is created by mutation in an Evolutionary Algorithm. The number of successive neighbourhoods examined before returning to the main EA loop is controlled by a *depth* parameter.

Within a memetic algorithm, one can consider the local search stage to occur as an improvement, or developmental learning phase within the evolutionary cycle, and it is a design choice whether the changes made to the individual (*acquired traits*) should be kept in the genotype (the Lamarckian paradigm), or whether the just resulting improved fitness should be awarded to the original (pre-local search) member of the population (the Baldwin paradigm). In a classic early study, Hinton and Nowlan [8] showed that the Baldwin effect could be used to improve the evolution of artificial neural networks, and a number of researchers have studied the relative benefits of Baldwinian versus Lamarckian algorithms. These two approaches both alter the fitness landscape:

- The Baldwin effect is to replace the fitness of each point with that of its fittest neighbour. To extend the landscape metaphor, this has the effect of broadening peaks and ridges, raising the height of valleys, and generally “blurring” the landscape structure and removing gradients and fine-grained structural features in a process similar to noise removal in image processing.

- The effect of Lamarckian learning is that the fitness of points in the landscape is unchanged, but a translation occurs to the higher neighbour, so that whole swathes of low-fitness points are effectively removed from the search space.

The aim of this paper is to examine whether the impact of these two different transformations is to reduce the size of the effect of different search strategies, and hence the information available to the self-adaptation process.

3 Experimental Methodology

3.1 Algorithm

The core EA used a very standard Genetic Algorithm (GA) following the parameter values suggested by previous authors. A (100,500) selection strategy with one point crossover (with probability 0.7), and bit-flipping mutation. Local search used a Hamming neighbourhood of distance one, with a greedy pivot rule accepting the first improvement, and depths of 0,1,2 or 5 successive neighbourhoods. Note that a local search depth of 0 equates to a standard GA.

The Self-adaptation process used the scheme outlined in [16,17,21]. Each solution encodes a choice from a discrete set of values, $1.0/l * \{0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 1.0, 2, 5, 10\}$ where l is the length of the problem encoding. Prior to mutating the solution encoding, the gene encoding for the mutation rate is randomly reset with probability $P_{sm} = 0.1$. Although these operators and parameter values were taken as fairly standard from the literature, preliminary experimentation (not show for reasons of space) suggests that the effects observed below occur over a wide range of parameter values. One point crossover was chosen for its positional bias which matches that of the problem encodings used.

3.2 Test Functions

The first set of problems were versions of the Royal Road fitness function [6]. In these the fitness is given by the number of blocks “aligned” to the target string (all 1s) in a problem with L blocks, each of length K . To examine the effect of learning as the size of the partitions (plateaus) increased, while keeping the size of the search space the same, 60 bit problems were used with $K \in \{2, 3, 4, 5, 6, 10, 15, 20\}$. A well known property of these functions is that for $K > 1$ they possess “plateaus” of equal fitness, that represent entropic barriers to evolutionary search. Search on these problems typically proceeds via a series of “epochs”. During transitions the entropy of the population is reduced as the correct alignment is found for the next block, and fixated through the population.

To understand the effect of learning on these problems, let us consider the partition of the search space corresponding to a single block. Applying one step of local search means that now K of the possible 2^K solutions in that partition now contribute to the global fitness instead of just 1. The effect of multiple steps of learning will depend on whether any of the blocks have unitation of $K - 1$. The “Baldwin effect” on these landscapes is that the plateaus effectively grow in size

to occupy a proportion $(K+1)/2^K$ of the partition. Regardless of mutation rate, it becomes more probable that mutation will cause a jump onto the plateau, but higher rates are more likely to destroy previously existing blocks, unless these can be repaired by multiple applications.

The effect of Lamarkian learning is subtly different - points with unitation in the partition between 0 and $K-2$ are unchanged, but those with unitation $K-1$ are removed as offspring created in those regions are moved to the single sub-solution with a unitation K . Thus the proportion of the partition corresponding to the high-fitness values is now $1/(2^k - K)$ which is smaller than the Baldwin version. Thus more of these points are at Hamming distance greater than 1, so we might expect to see the selection of higher mutation rates which are more likely to cause jumps to points at distance 1 from the optimal sub-solution.

The second class of static problems are deceptive ones, that present a fitness barrier, rather than an entropic one, to evolutionary progress to the global optimum. These so-called L "Trap" or deceptive functions of size K . This paper will consider functions composed L contiguous sub-functions. Each of these is a deceptive partition of size K bits, where the reward was $100/L$ for all 1s, otherwise $0.88 * (K - u(i))/L$ where $u(i)$ is the unitation in the i^{th} partition. Again we used 60 bit problems and the same set of values for K .

The final problem is used to explore the interaction between learning, and self-adaptation's well known ability to respond automatically to changes in the fitness landscape. Hence the third test problem used is a 200-bit variant of the unimodal OneMax function, switching to the opposite (ZeroMax) after 25 generations: Before the landscape shift, the effect of Baldwinian learning with depth d on this landscape is to assign to each genome the fitness of a individual with d more bits set to 1 - in other words the shape of the landscape is left untouched except for those few solutions with a neighbourhood $H(i, j) = d$ of the global optimum, where the landscape is flat. The effect of Lamarkian search is to move each point d steps up the slope of the hill - ie. effectively to remove those points with $u(i) < d$ from the search space. In both cases the underlying structure of the problem is left unchanged, so except for the more rapid convergence to the global optimum, it is hypothesized that the self-adaptation of mutation rates will follow a similar pattern to the GA.

These values of length used were chosen to provide similar levels and speed of convergence for each problem given the selection regime and population sizes.

3.3 Methods for Analysis

Each configuration of EA without local search (GA), and with Baldwin (B) or Lamarkian (L) learning with depths 1, 2 and 5 (B-d1, ..., B-d5, L-d1 etc.) was run 100 times on each problem, with a termination criteria of 50 generations. After each generation of each run data was recorded for the best, worst and mean fitness, mean and standard deviation of mutation rates in the current population, and the total number of evaluations used.

As this paper is primarily concerned with the effect on the learning of mutation rates, algorithms are compared generation-by-generation, ignoring the fact that the local search variants make more calls to the evaluation function.

In separate experiments the mean best fitness, average evaluations to solution and success rates were compared for the seven algorithms above, and variants using a fixed mutation rate of $pm = 1/l$. Where appropriate, algorithms have been compared using statistical analysis - either at snapshots of specific generations, or averaged over the whole runs. We used SPSS v20 to conduct ANalysis of VAriance (ANOVA) followed by appropriate post-hoc testing to look for “homogenous subsets” which fail pairwise tests for statistically significant differences at the 95% confidence level. Results shown in the form $A < \{B, C\} < \{C, D\}$ mean that values for set A are significantly lower than those for sets B, C and D . Values for B are not significantly lower than those for C but are for D .

4 Results

4.1 Benchmarking Self-Adaptation

Comparing effectiveness, by pooling results and performing ANOVA on the maximum fitnesses, with the function and algorithm as independent factors showed that although there were small differences between algorithms, by 49 generations there were no statistically significant differences between fixed and self-adaptive mutation rates. Comparing the final mean mutation rates, those of the MA-B-d5 algorithm were significantly higher than the other methods, which were otherwise not significantly different.

Comparing the efficiency, as measured by when the best fitness was recorded for each run, showed that the self-adaptive variants were always faster, more significantly so with increased depth of local search. Lamarkian variants were always significantly faster than their Baldwinian counterparts and increase of depth from 0 (GA) through to 5 caused a significant increase in evaluations.

The mean best fitness results showed that there was no difference between the fixed and adaptive mutation rates for Lamarkian search, but these were always significantly better than the GA and Baldwinian MAs. In contrast, adding self-adaptation to the Baldwinian MAs significantly reduced the mean best fitness for each different depth of search.

4.2 Analysis of Evolved Behaviours on Different Functions

The next set of experiments concentrate on the effect of selection at the level of mutation rates in the presence of different forms of local search. To this end, the “strategy adaptation” parameter P_{sm} was set to 0, so each member of the initial population had its mutation rate randomly set to one of the permissible values, and offspring inherited mutation rates unchanged from their parents. The results are shown graphically in Figure 1, which shows how the patterns of the evolved behaviour change between problems and algorithms. Five characteristics of the population (best, mean, worst fitness, mean and standard deviation of mutation rate) (y -axis) are plotted against the number of generations (x -axis).

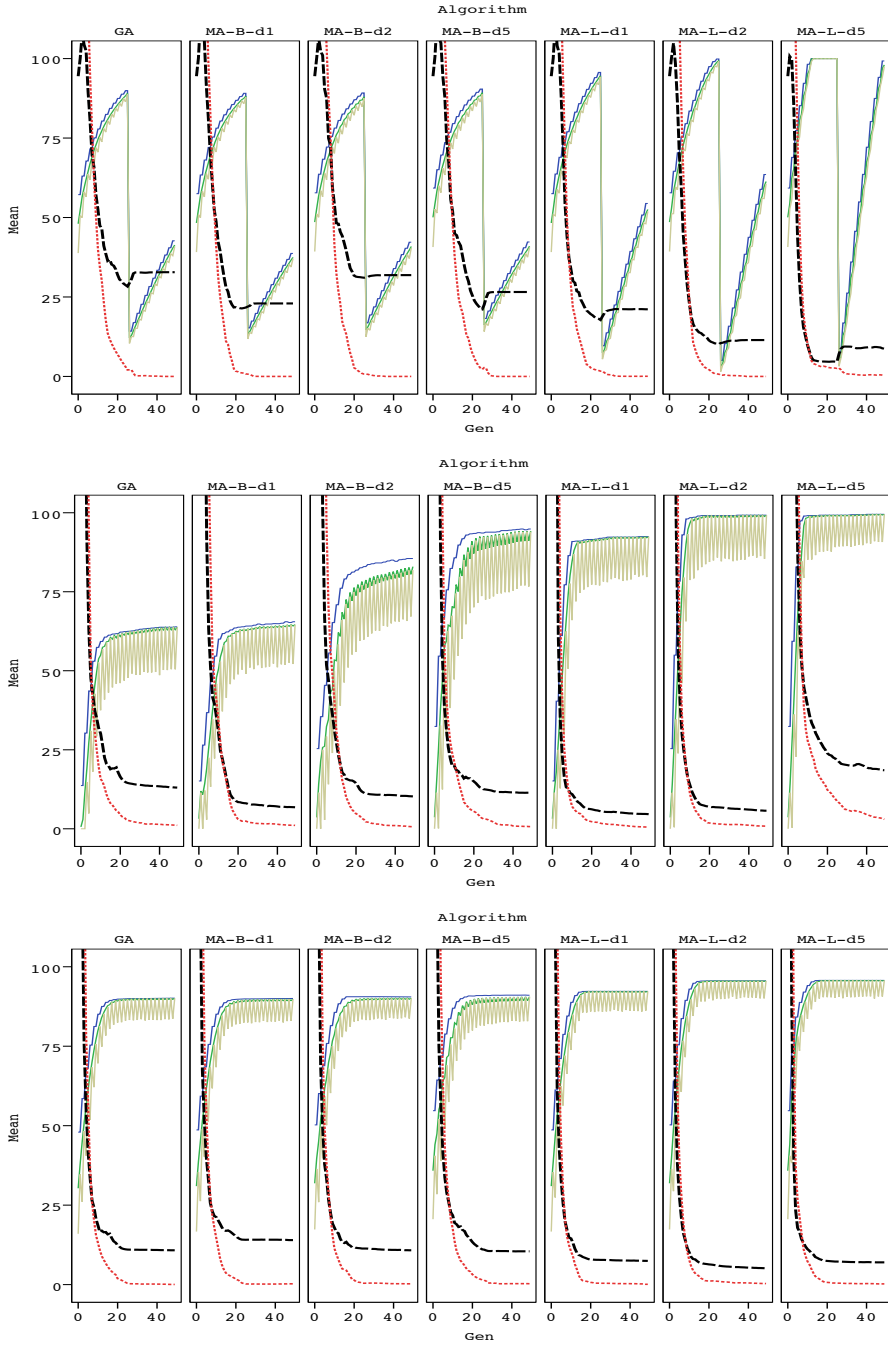


Fig. 1. Illustrative Evolving Behaviour: population best, mean and worst fitness (out of 100), mean (dashed line) and standard deviation(dotted) of mutation rates (probability x 10000). Switcher (top), Royal Road with K=8 (middle) and Trap with K=4 (bottom).

Evolution of Mutation Rates for OneMax. For both paradigms the mutation rates stabilised more slowly, and to values that decrease with increasing search depth. However for Baldwinian search, the values at generation 49 are not significantly different to the GA. The effect of selection is much more noticeable with Lamarkian learning. The mutation rates converge faster, and to lower values than the GA - not significantly so for depth 1, but the evolved rates for depths 2 and 5 are significantly different to the GA, and each other. The reduction in the standard deviation shows that this is a learned effect rather than simple drift. To confirm this, experiments were run where the function switched from OneMax to Zeromax after 25 generations. Figure 1 (top) shows a clear spike in the mutation rates after the switch. The subsequent rapid recovery in fitness, most notably for MA-L-D5, is evidence of effective self-adaption.

Results for Royal Road Functions. Figure 1 (middle) shows the evolution of behaviour on the Royal Road function with partitions of size 8. In addition to the difference in effectiveness of search, the key point to note is the consistently higher, and more varied mutation rates for Lamarkian search with depth 5, a feature that increases when the size of the sub-blocks to be optimised increased. Mutation rates also increase with depth of Baldwinian search, but the differences are not significant by generation 49

Results for Deceptive Functions. Figure 1 (bottom) shows the evolution of behaviour on the deceptive function with blocks of size 4. Note the difference in effectiveness of search. On both functions, at generation 49 the statistically homogenous subsets are, ranked according to increasing fitness; (B-d5, GA, B-d1, B-d2) < L-d1 < (L-d5, L-d2), where the suffix MA is omitted for brevity.

On the functions with 4-bit partitions, the Baldwin behaviour is not statistically significantly different to the GA, but there are consistently lower mutation rates for the Lamarkian learning. This difference is significant even up to generation 49 when the best value had stopped increasing. With the trap-8 function, the values are no longer statistically significant by generation 49 - but of course there are far fewer sub-functions to be optimised. Considering instead the mean mutation rates across the whole run, there is now a statistically significant difference - the values for Lamarkian learning are significantly lower than for the GA, and then in turn for the Baldwinian learning. These values reflect the speed of the adaptive process- higher mean values meaning slower adaptation.

5 Discussion

The first set of benchmarking comparisons confirmed that self-adaptation outperformed a single fixed mutation rate, as expected - working just as effectively at finding good solutions but more efficiently. Lamarkian learning improved the mean best fitness discovered. However, the interplay between the Baldwin effect and self-adaptation was not always beneficial - particular on the Royal Road landscapes where the plateaus form entropic barriers to improvement and the Baldwin effect extends those plateaus.

On the OneMax function, the hypothesis predicted that Lamarkian learning would demonstrate faster adaptation (H2) and to lower (H3) values of mutation rates than the GA. This was supported by the observations. The hypothesis H1 and H3 suggested competing effects would result from Baldwinian learning. Results confirmed that and indeed with depth 1 a slower adaptation to higher rates than the GA was seen, an effect which diminished with increased local search depth, but the differences were not statistically significant by the end of even these relatively brief runs.

The results on the switcher function confirmed that self-adaptation is able to occur effectively and efficiently with Lamarkian learning up to a depth 5, possibly even suggesting a synergistic effect when compared to the GA alone.

On the Royal Road functions the hypothesised effects were not really seen except for with depth 5, where as predicted by H4, the Lamarkian search maintains higher mutation rates - which in turn lead to the continued discovery of sub-solutions. For example even after averaging over 100 runs, the middle right figure of Figure 1 shows an increase in f_{max} around 30 generations.

On the trap functions the differences are most evident in the speed of adaptation: as predicted by H1 the “blurring” effect of Baldwin learning significantly reduces the rate of adaptation to lower mutation values than the GA. In contrast, as predicted by H2, the rate of adaptation is faster for Lamarkian learning than for the GA, and hence the overall mean across all generations is lower.

6 Conclusions

This paper set out to examine the interaction between two different forms of memetic learning, and the self-adaptation of mutation rates. The primary empirical results suggest that whereas Lamarkian learning seems to reinforce self-adaptation, the Baldwin effect often hinders the process, sometimes with detrimental results on the effectiveness and efficiency of the overall search. The message of this paper is therefore perhaps unsurprising: that it is unwise to rashly mix algorithmic adaptations that work well in isolation. Clearly further studies are needed to model these effects so that the twin forces of memetics and self-adaptation can be brought to bear with reliable and predictable results.

References

1. 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)
2. Bäck, T.: Self adaptation in genetic algorithms. In: Varela, F., Bourgine, P. (eds.) Toward a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life, pp. 263–271. MIT Press, Cambridge (1992)
3. Beyer, H.-G.: The Theory of Evolution Strategies. Springer, New York (2001)

4. Eiben, A., Michalewicz, Z., Schoenauer, M., Smith, J.: Parameter Control in Evolutionary Algorithms. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) *Parameter Setting in Evolutionary Algorithms*. SCI, vol. 54, pp. 19–46. Springer, Heidelberg (2007)
5. Eiben, A., Smith, J.: *Introduction to Evolutionary Computation*. Springer (2003)
6. Forrest, S., Mitchell, M.: Relative building block fitness and the building block hypothesis. In: Whitley, L. (ed.) *Foundations of Genetic Algorithms 2*, pp. 109–126. Morgan Kaufmann, San Francisco (1992)
7. 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)
8. Hinton, G., Nowlan, S.: How learning can guide evolution. *Complex Systems* 1, 495–502 (1987)
9. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: Model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computation* 9(5), 474–488 (2005)
10. Meyer-Nieberg, S., Beyer, H.-G.: Self-Adaptation in Evolutionary Algorithms. In: *Parameter setting in evolutionary algorithms*, pp. 47–75. Springer, Heidelberg (2007)
11. Ong, Y.S., Lim, M.H., Chen, X.: Memetic Computation—Past, Present & Future. *IEEE Computational Intelligence Magazine* 5(2), 24–31 (2010)
12. Preuss, M., Bartz-Beielstein, T.: Sequential parameter optimization applied to self-adaptation for binary-coded evolutionary algorithms. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) *Parameter Setting in Evolutionary Algorithms*. SCI, vol. 54, pp. 91–119. Springer, Heidelberg (2007)
13. Rudolph, G.: Self-adaptive mutations lead to premature convergence. *IEEE Transactions on Evolutionary Computation* 5, 410–414 (2001)
14. Schwefel, H.-P.: *Numerical Optimisation of Computer Models*. Wiley (1981)
15. Serpell, M., Smith, J.: Self-adaption of mutation operator and probability for permutation representations in genetic algorithms. *Evolutionary Computation* 18(3), 491–514 (2010)
16. Smith, J.: Modelling GAs with self-adaptive mutation rates. In: Spector, L., et al. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 599–606. Morgan Kaufmann, San Francisco (2001)
17. Smith, J.: Parameter perturbation mechanisms in binary coded gas with self-adaptive mutation. In: Rowe, Poli, DeJong, Cotta (eds.) *Foundations of Genetic Algorithms 7*, pp. 329–346. Morgan Kaufmann, San Francisco (2003)
18. Smith, J.: Co-evolving memetic algorithms: A review and progress report. *IEEE Transactions in Systems, Man and Cybernetics, Part B* 37(1), 6–17 (2007)
19. Smith, J.: Estimating meme fitness in adaptive memetic algorithms for combinatorial problems. *Evolutionary Computation* 20(2), 165–188 (2012)
20. Smith, J., Fogarty, T.: Self adaptation of mutation rates in a steady state genetic algorithm. In: *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*, pp. 318–323. IEEE Press, Piscataway (1996)
21. Stone, C., Smith, J.: Strategy parameter variety in self-adaption. In: Langdon, W., et al. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, 9-13 July, pp. 586–593. Morgan Kaufmann, San Francisco (2002)

On Low Complexity Acceleration Techniques for Randomized Optimization

Sebastian Urban Stich

Institute of Theoretical Computer Science
ETH Zürich, 8092 Zürich, Switzerland
sstich@inf.ethz.ch

Abstract. Recently it was shown by Nesterov (2011) that techniques from convex optimization can be used to successfully accelerate simple derivative-free randomized optimization methods. The appeal of those schemes lies in their low complexity, which is only $\Theta(n)$ per iteration—compared to $\Theta(n^2)$ for algorithms storing second-order information or covariance matrices. From a high-level point of view, those accelerated schemes employ correlations between successive iterates—a concept looking similar to the evolution path used in Covariance Matrix Adaptation Evolution Strategies (CMA-ES). In this contribution, we (i) implement and empirically test a simple accelerated random search scheme (SARP). Our study is the first to provide numerical evidence that SARP can effectively be implemented with adaptive step size control and does not require access to gradient or advanced line search oracles. We (ii) try to empirically verify the supposed analogy between the evolution path and SARP. We propose an algorithm CMA-EP that uses only the evolution path to bias the search. This algorithm can be generalized to a family of low memory schemes, with complexity $\Theta(mn)$ per iteration, following a recent approach by Loshchilov (2014). The study shows that the performance of CMA-EP heavily depends on the spectra of the objective function and thus it cannot accelerate as consistently as SARP.

Keywords: Gradient-free optimization, accelerated random search, evolution path, adaptive step size, Covariance Matrix Adaptation, spectra.

1 Introduction

The Gradient Method [1, 2]—one of the most fundamental schemes in convex optimization—has iteration complexity $\Theta(n)$, where n is the dimension. On strongly convex functions its convergence rate is linear, depending only on the condition number of the objective function. To overcome the difficulty imposed by ill-conditioned problems, second-order methods like Newton’s method or first order Quasi-Newton methods such as the BFGS scheme [3–5] are a welcome alternative. Those schemes maintain a quadratic model of the objective function and their complexity is bounded by $\Omega(n^2)$. Limited memory schemes like L-BFGS [6, 7] trade-off linear iteration complexity $\Theta(mn)$ (where m is a fixed parameter), versus convergence rate. Accelerated versions of the Gradient Method

have linear complexity $\Theta(n)$ per iteration and converge with optimal rate among all first-order methods. On strongly convex problems the convergence rate is proportional to the square root of the condition number [1, 2, 8–10].

Randomized (gradient-free) schemes do not require first-order information, they operate by only querying function values. Such schemes are nowadays a ubiquitous tool for solving many practical problems in science and engineering where first-order information is difficult to compute or does not exist. Among the first proposed schemes that are still of considerable (theoretical) importance are Adaptive Step Size Random Search (aSS) [11] and the (almost identical) well-known (1+1)-Evolution Strategy (ES) [12] in Evolutionary Computation. More recent schemes comprise Random Pursuit (RP) [13, 14], or Random Gradient Descent [15]. Those schemes can be viewed as generalizations of the Gradient Method to zeroth-order, with iteration complexity $\Theta(n)$. Likewise, analogues of the second-order schemes try to estimate an approximation of the Hessian by finite difference computations [16, 17] or by estimating correlations among search directions. A very popular algorithm of this kind is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [18, 19]. Limited memory variants have been proposed in [20, 21], with iteration complexity $\Theta(mn)$. Especially, the later variant due to Loshchilov shows excellent convergence in high dimensions also for small values of m . Zeroth-order analogues of the accelerated gradient schemes have been introduced [15, 22]. Those schemes massively outperform the simple random search schemes on convex problems. This performance gain does not come for free, as those schemes require valid bounds on the condition number as input parameters. However, their low iteration complexity of $\Theta(n)$ could make them a promising choice for large scale problems, where the fully-quadratic schemes inherently fail. We focus on a very simple accelerated random search scheme, which we call SARP.

By inspecting closely the accelerated search schemes, one could conclude that the difference to the classical schemes can be explained by an additional “drift” term [2, 10] that takes into account correlations of the last iterates. In the popular CMA-ES, correlations between successive iterates are accumulated in the evolution path [23]. In this work, we are interested, if the evolution path can be used for acceleration, competitive to the accelerated zeroth-order schemes for convex optimization. To this end, we introduce a variant of CMA-ES, called EP-CMA, that only uses the information stored in the evolution path to bias the direction of the search. Similar to the approach proposed in [21], this scheme can be generalized to a family of schemes, which we call EP-CMA- m . However, we do not present an efficient, i.e. $\Theta(mn)$, implementation of those schemes as this was not required here with dimension $n \leq 100$ in our empirical study.

The remainder of this paper is structured as follows. In Section 2 we present the accelerated random search scheme SARP and detail the EP-CMA- m schemes. In Section 3 we empirically test the performance of all schemes on three quadratic and the non-convex Rosenbrock function, and highlight the key results. We discuss these results and conclude the paper in Section 4.

lineSearch($\mathbf{x}, \mathbf{u}, [\sigma, p]$)	aSS($\mathbf{x}, \mathbf{u}, \sigma, p$) (<i>adaptive step size</i>)
1 if <i>exact</i> then return exactLS($\mathbf{x}, \mathbf{u}/\ \mathbf{u}\ $)	1 if $f(\mathbf{x} + \sigma\mathbf{u}) \leq f(\mathbf{x})$ then
2 else return aSS($\mathbf{x}, \mathbf{u}, \sigma, p$)	2 $\mathbf{x}_+ \leftarrow \mathbf{x} + \sigma\mathbf{u}; \sigma_+ \leftarrow \sigma \cdot \exp(1/3)$
exactLS($\mathbf{x}, \mathbf{u}, [\sigma]$)	else
1 $\sigma_+ \leftarrow \min_{\lambda} f(\mathbf{x} + \lambda\mathbf{u}); \mathbf{x}_+ \leftarrow \mathbf{x} + \sigma_+\mathbf{u}$	3 $\mathbf{x}_+ \leftarrow \mathbf{x}; \sigma_+ \leftarrow \sigma \cdot \exp\left(-\frac{p}{3(1-p)}\right)$
2 return (\mathbf{x}_+, σ_+)	4 return (\mathbf{x}_+, σ_+)

Fig. 1. Line search oracles for gradient-free optimization

2 Algorithms

We here present the optimization schemes considered in this study. We first detail two Random Pursuit algorithms and a simplistic variant of a standard (1+1)-CMA-ES. Then we introduce the new EP-CMA- m schemes.

RP. Random Pursuit is a basic optimization scheme that iteratively generates a sequence of approximate solutions to the global optimization problem $\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$. In each step a search direction is drawn $\mathbf{u}_k \sim \mathcal{N}(0, \mathbf{1}_n)$. In Random Pursuit with exact line search (RP-exact), first proposed in [13] and analyzed in [14], the step size σ is determined by minimizing the objective function in direction \mathbf{u} , i.e. $\sigma = \arg \min_{\lambda} f(\mathbf{x} + \lambda\mathbf{u})$. For quadratic functions $f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T A \mathbf{x}$ with Hessian A , the expected one-step progress can be estimated as:

$$\mathbb{E}[f(\mathbf{x}_+) \mid \mathbf{x}] \leq (1 - \lambda_{\min}(A)/\text{Tr}[A]) f(\mathbf{x}), \quad (1)$$

where \mathbf{x} is the current iterate, $\mathbf{x}_+ := \mathbf{x} + \sigma\mathbf{u}$ denotes the next iterate. This statement can also be generalized to arbitrary smooth convex functions [14]. Stich et al. [14] show that RP-exact still converges if the line search is not performed exactly, but allowing relative errors. Therefore, we also consider Random Pursuit with adaptive step sizes (RP) instead of exact line search. In RP the step size is dynamically controlled such as to approximately guarantee a certain probability p of finding an improving iterate. Depending on the underlying test function, different optimality conditions can be formulated for the value p . Schumer and Steiglitz [11] suggest the setting $p = 0.27$ which is considered throughout this work. We use immediate exponential step size control as explicitly formulated in the aSS sub-routine in Fig. 1. RP is identical to the well known (1+1)-ES.

SARP. Accelerated random search schemes are fundamentally different from the simple random search schemes. Instead of generating only one sequence of iterates, those algorithms typically maintain two or more sequences simultaneously (here essentially \mathbf{x}_k and \mathbf{y}_k , see Fig. 2). Those sequences allow to store gathered knowledge on the objective function which yields better performance. In Fig. 2 we present a simple version of the accelerated random search scheme proposed in [14] and refer to it as simple accelerated random search (SARP). Like RP, SARP can (in practice) be used with exact line search oracles or with adaptive step size control, although convergence for those oracles has not been proven yet. For Nesterov’s accelerated random search scheme [15], the expected one-step progress can be estimated as

<hr/> <p style="text-align: center;">RP($\mathbf{x}_0, N, [\sigma_0, p]$)</p> <hr/> <pre> 1 for $k = 1$ to N do 2 $\mathbf{u}_k \sim \mathcal{N}(0, I_n)$ 3 $(\mathbf{x}_k, \sigma_k) \leftarrow \text{lineSearch}(\mathbf{x}_{k-1}, \mathbf{u}_k, [\sigma_{k-1}])$ 4 return \mathbf{x}_N </pre> <hr/> <p style="text-align: center;">EP-CMA-m ($\mathbf{x}_0, N, \sigma_0, p, c_c, c_{\text{cov}}$)</p> <hr/> <pre> 1 $\hat{\mathbf{p}}_0 \leftarrow \mathbf{0}$; $\hat{\mathbf{p}}_1, \dots, \hat{\mathbf{p}}_{m-1} \leftarrow \mathbf{0}$; $q = 0$ 2 for $k = 1$ to N do 3 $C_k \leftarrow I_n$ 4 for $i = 1$ to $m-1$ do 5 $C_k \leftarrow (1 - c_{\text{cov}})C_k + c_{\text{cov}}\hat{\mathbf{p}}_i \hat{\mathbf{p}}_i^T$ 6 $\mathbf{u}_k \sim \mathcal{N}(0, C_k)$ 7 $(\mathbf{x}_k, \sigma_k) \leftarrow \text{aSS}(\mathbf{x}_{k-1}, \mathbf{u}_k, \sigma_{k-1})$ 8 $\mathbf{y}_k \leftarrow (\mathbf{x}_k - \mathbf{x}_{k-1})/\sigma_{k-1}$ 9 if $\mathbf{y}_k \neq \mathbf{0}$ (success) then 10 $\mathbf{p}_k \leftarrow (1 - c_c)\mathbf{p}_{k-1} + \sqrt{c_c(2 - c_c)}\mathbf{y}_k$ 11 else $\mathbf{p}_k \leftarrow (1 - c_p)\mathbf{p}_{k-1}$ if $k > q + n^2/m$ 12 then 13 $\hat{\mathbf{p}}_1 \leftarrow \hat{\mathbf{p}}_2, \dots, \hat{\mathbf{p}}_{m-2} \leftarrow \hat{\mathbf{p}}_{m-1}$; $q = k$ 13 return \mathbf{x}_N </pre> <hr/>	<hr/> <p style="text-align: center;">SARP($\mathbf{x}_0, N, m, L, [\sigma_0]$)</p> <hr/> <pre> 1 $\mathbf{y}_0 \leftarrow \mathbf{x}_0$; $\mathbf{v}_0 \leftarrow \mathbf{x}_0$; $\theta \leftarrow \sqrt{\frac{m}{2n^2L}}$ 2 for $k = 1$ to N do 3 $\mathbf{u}_k \sim \mathcal{N}(0, I_n)$ 4 $(\mathbf{x}_k, \sigma_k) \leftarrow \text{lineSearch}(\mathbf{y}_{k-1}, \mathbf{u}_k, [\sigma_{k-1}])$ 5 $\mathbf{y}_k \leftarrow (\theta\mathbf{v}_{k-1} + \mathbf{x}_k)/(1 + \theta)$ 6 $\mathbf{v}_k \leftarrow (1 - \theta)\mathbf{v}_{k-1} + \theta\mathbf{y}_k + \theta n \frac{L}{m} \sigma_k \mathbf{u}_k$ 7 return \mathbf{x}_N </pre> <hr/> <p style="text-align: center;">(1+1)-CMA($\mathbf{x}_0, N, \sigma_0, p, c_c, c_{\text{cov}}$)</p> <hr/> <pre> 1 $C_0 \leftarrow I_n$; $\mathbf{p}_0 \leftarrow \mathbf{0}$ 2 for $k = 1$ to N do 3 $\mathbf{u}_k \sim \mathcal{N}(0, C_{k-1})$ 4 $(\mathbf{x}_k, \sigma_k) \leftarrow \text{aSS}(\mathbf{x}_{k-1}, \mathbf{u}_k, \sigma_{k-1})$ 5 $\mathbf{y}_k \leftarrow (\mathbf{x}_k - \mathbf{x}_{k-1})/\sigma_{k-1}$ 6 if $\mathbf{y}_k \neq \mathbf{0}$ (success) then 7 $\mathbf{p}_k \leftarrow (1 - c_c)\mathbf{p}_{k-1} + \sqrt{c_c(2 - c_c)}\mathbf{y}_k$ 8 $C_k \leftarrow (1 - c_{\text{cov}})C_{k-1} + c_{\text{cov}}\mathbf{p}_k \mathbf{p}_k^T$ 9 else 10 $C_k \leftarrow C_{k-1}$; $\mathbf{p}_k \leftarrow (1 - c_p)\mathbf{p}_{k-1}$ 10 return \mathbf{x}_N </pre> <hr/>
--	---

Fig. 2. RP, EP-CMA and CMA-ES schemes

$$\mathbb{E}[f(\mathbf{x}_+) | \mathbf{x}] \leq (1 - (n\sqrt{\kappa})^{-1}) f(\mathbf{x}), \quad (2)$$

where condition $\kappa = L/m$ and the two parameters $m \leq \lambda_{\min}(A)$ and $L \geq \lambda_{\max}(A)$ are required as input to the algorithm (and always provided in our numerical study). This rate is much better than (1) and we hope to see that SARP attains comparable performance. SARP is not a monotone scheme, that is, the function values of the iterates are not monotonically decreasing. SARP is closely related to the first-order accelerated search scheme of Nesterov [2]. This scheme also simultaneously maintains two sequences \mathbf{x}'_k and \mathbf{y}'_k of iterates (but requires access to the gradient in every iteration). For Nesterov's first-order scheme it is known [2, p.79] that the sequence \mathbf{y}'_k obeys

$$\mathbf{y}'_{k+1} = \mathbf{x}'_{k+1} + \beta' (\mathbf{x}'_{k+1} - \mathbf{x}'_k), \quad (3)$$

for $\beta' = 1 - 2/\sqrt{\kappa} + O(1/\kappa)$. Thus the additional $(\mathbf{x}'_{k+1} - \mathbf{x}'_k)$ acts like a drift term, cf. [1]. For SARP with parameter $\theta' = \sqrt{1/(n^2\kappa)}$ (only slightly different from θ in Fig. 2) the same reformulation of the update reveals

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \beta (\mathbf{x}_{k+1} - \mathbf{x}_k), \quad (4)$$

for $\beta = (1 - \theta')/(1 + \theta') = 1 - 2/(n\sqrt{\kappa}) + O(1/\kappa)$. The main term contributing to the drift is approximately only an $1/n$ -fraction of the step, accounting for the uncertainty emerging from the randomness.

(1+1)-CMA-ES. In contrast to the presented Random Pursuit schemes, in CMA-ES new search points are sampled from a multivariate normal distribution

$\mathbf{u}_k \sim \mathcal{N}(0, C_k)$ whose parameter C_k is updated in each iteration based on the evaluation of the samples. The covariance matrix can be adapted using different rank-1 [18, 24] or rank-k updates [19]. In addition, the CMA-ES scheme is augmented by an auxiliary variable called evolution path that takes into account the correlation of successive means taken over a finite horizon. In [18, 23], the evolution path \mathbf{p}_k is updated as

$$\mathbf{p}_{k+1} = (1 - c_c)\mathbf{p}_k + \sqrt{c_c(1 - c_c)}\mathbf{u}_k. \quad (5)$$

Cumulative information about successive steps is stored in the variable \mathbf{p}_k . We use a simplistic CMA-ES variant, closely following [18], see Fig. 2. We use the simple Adaptive Step Size control aSS to determine the step size σ_k , the covariance matrix update solely uses the information of the evolution path like in [24] and for simplicity we refrain from implementing any regularization features, in contrast to [24]. We use the same parameters that were proposed in [24] for the (1+1)-CMA-ES, namely $c_c = 2/(n + 2)$, $c_p = 1/12$ and $c_{\text{cov}} = 2/(n^2 + 6)$.

EP-CMA-1. The evolution path \mathbf{p}_k accumulates information over successful steps. This accumulation can be seen as a smoothing of the noisy information obtained in single steps, at the effect that the evolution path points into direction of more promising function values [18]. In this study, we are interested if the evolution path can be used in a similar way as the drift term in (3) or (4), respectively, to accelerate the search. There are several ways to incorporate the evolution path \mathbf{p}_k into the update scheme. We suggest to use the path \mathbf{p}_k in the following way: in the simple random search scheme RP (equivalent to (1+1)-ES), we sample in iteration k a direction from $\mathbf{u}_k \sim \mathcal{N}(0, (1 - c_{\text{cov}})I_n + c_{\text{cov}}\mathbf{p}_k\mathbf{p}_k^T)$, with bias along the direction indicated by \mathbf{p}_k . This has the effect that we only follow successful steps, but the drift imposed by the evolution path might be smaller than it ideally should be. The scheme EP-CMA-1 is detailed in Fig 2, we used c_c and c_p as above, and $c_{\text{cov}} = 1/5$. This approach is similar to [25].

EP-CMA- m . The proposed EP-CMA-1 can easily be generalized to a whole family of optimization schemes by an approach presented in [21]. In EP-CMA-1, only the information stored in the current evolution path \mathbf{p}_k is used to bias the search direction. But we could also afford to temporarily store a small number m of past $\mathbf{p}_{k'}$ for $k' < k$, and use the information collectively to bias the search. As two successive evolution paths are likely highly correlated, we propose to store the evolution path only every n^2/m -th generation (and up to at most $(m - 1)$ copies simultaneously). The resulting scheme is detailed in Fig. 2. We used $c_c = 2/(n + 2)$ as in CMA-ES, and for $m > 1$, $c_{\text{cov}} = 2/(6 + m)$ for EP-CMA- m . If implemented carefully, EP-CMA- m has $\Theta(mn)$ complexity per iteration (not shown in Fig 2). For $m = n^2$, the updates of EP-CMA- m are identical to the updates of (1+1)-CMA-ES, if *limited* to a finite horizon of n^2 steps. In contrast, the low memory method proposed in [20] behaves similar to CMA-ES already for $m = n$, but has iteration complexity $\Theta(nm^2)$.

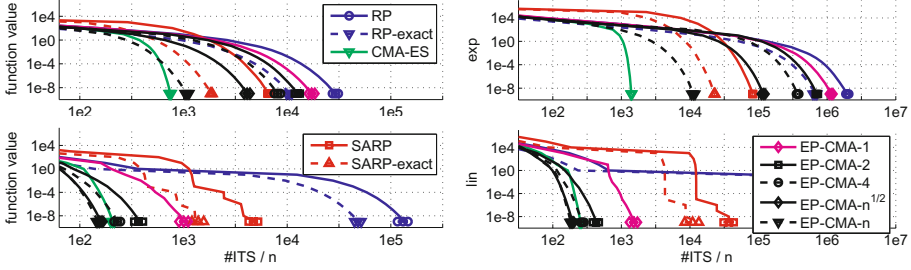


Fig. 3. Evolution of FVAL vs. #ITS on f_{exp} (top) and f_{lin} (bottom) with $L = 1E4$ (left) and $L = 1E6$ (right) in $n = 100$ dimensions. For 51 (11 for RP on f_{lin} with $L = 1E6$) runs we recorded #ITS needed to reach FVAL of $1E-9$. The trajectory realizing the median values is depicted, mean and one standard deviation are indicated by markers. (RP on f_{lin} with $L = 1E6$ reaching FVAL $< 1E-2$ after $1E6.5n$ #ITS.)

3 Empirical Study

We now present the setup of our empirical study. We focus on the following schemes: (i) the two Random Pursuit schemes with adaptive step size control (denoted as RP and SARP) and with exact line search (denoted as RP-exact and SARP-exact), (ii) the simplified (1+1)-CMA-ES and (iii) the EP-CMA- m schemes as introduced in Sec. 2, see Fig. 2. We use EP-CMA- m with parameters $m = 1, 2, 4, \sqrt{n}, n$. This totals in 10 different schemes, all of which were implemented in MATLAB and will be made available on the authors website.

We tested the performance of all algorithms on three variants of the ellipsoidal benchmark function [18] and the non-convex Rosenbrock function, detailed in Table 1. The quadratic functions were chosen in such a way that the extremal values of their spectra (1 and L) both agree. We considered the quadratic functions with parameters $L = 1E4$ and $L = 1E6$ each, and repeat the experiments in dimensions $n = 20, 40, 60, 80, 100$.

For all experiments, initial settings were $\mathbf{x}_0 = \mathbf{1}$, $\sigma_0 = 1$ and $p = 0.27$ (for schemes with the aSS routine). We count the number of iterations (#ITS) needed to decrease the function value (FVAL) below $1E-9$. A graphical summary of our results can be found in Fig. 3-5. Results not depicted here are reported in the supplementary online material [26]. We now proceed by discussing some of the key results.

Table 1. List of benchmark functions

$f_{\text{exp}}(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n L^{\frac{i-1}{n-1}} x_i^2$	$f_{\text{rosen}}(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right)$
$f_{\text{lin}}(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n \left(1 + i \frac{(L-1)}{(n-1)} \right) x_i^2$	$f_{\text{two}}(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{\lfloor n/2 \rfloor} x_i^2 + \frac{L}{2} \sum_{i=\lfloor n/2 \rfloor}^n x_i^2$

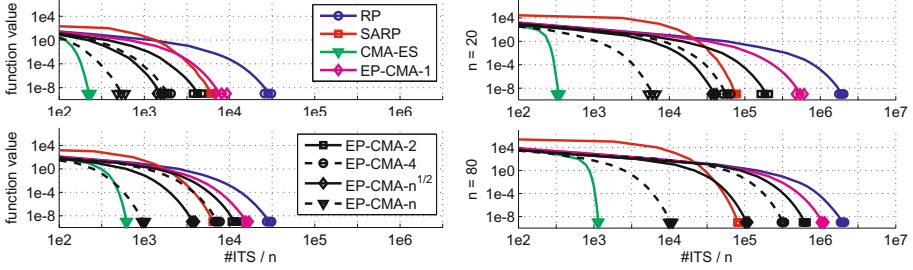


Fig. 4. Evolution of FVAL vs. #ITS on f_{exp} with $L = 1\text{E}4$ (left) and $L = 1\text{E}6$ (right) in $n = 20$ and $n = 80$ dimensions. For 51 runs we recorded #ITS needed to reach FVAL of $1\text{E}-9$. The trajectory realizing the median values is depicted, mean and one standard deviation are indicated by markers.

Line Search. Both RP and SARP were tested with exact line search oracle and adaptive step size control. In Fig. 3 we see that the exact schemes outperform their adaptive variants in $n = 100$ dimensions by a factor of roughly 2-3. This pattern is observed throughout the whole benchmark in all dimensions. Thus we omit to display the results for exact line search in subsequent Figs. 4-5.

SARP vs. EP-CMA-1. The picture is twofold. In Fig. 3 we see that EP-CMA-1 outperforms SARP by a factor of roughly 5 on f_{lin} with $L = 1\text{E}4$ (factor 24 for $L = 1\text{E}6$). The smallest eigenvalue of this function is separated from the second largest by a gap of roughly n . Hence, knowledge of one important direction reduces the conditioning of the function by a large factor. This factor becomes smaller in higher dimension. This scaling in the dimensions is indeed observed empirically, and depicted in [26].

On the other three functions, SARP performs consistently better than EP-CMA-1. On f_{exp} with $L = 1\text{E}4$ in $n = 100$ dimensions the factor is roughly 3, its roughly 14 for $L = 1\text{E}6$ (Fig. 3), and exceeds 10 on both f_{two} and f_{rosen} (Fig. 5). Considering the scaling in dimension (Figs. 4-5; and [26]), we observe that the relative performance ($\#ITS/n$) of SARP remains constant on all four benchmark functions, as predicted by theory for a similar method [15, 22].

EP-CMA-schemes. The EP-CMA- m schemes consistently work better for increasing values of m throughout the whole benchmark (Figs. 3-5). On f_{exp} with $L = 1\text{E}4$ the difference in #ITS between EP-CMA- n and EP-CMA-1 is roughly a factor of 10, and 20 for $L = 1\text{E}6$ (Fig. 3). The gap becomes gradually smaller on f_{lin} , f_{two} (especially for $L = 1\text{E}6$, see [26]), and is insignificant on f_{rosen} (Fig. 5). On f_{lin} the EP-CMA- m schemes perform extremely well, already for small m . EP-CMA-4 performs approximately as good as CMA-ES, for both parameters $L = 1\text{E}4$ and $L = 1\text{E}6$ (Fig. 3). On f_{exp} in $n = 100$ dimensions and parameter $L = 1\text{E}4$, both SARP and EP-CMA-4 need about the same #ITS. For parameter $L = 1\text{E}4$ the performance of SARP is the same as the performance of EP-CMA- \sqrt{n} (Fig. 3). On both f_{two} and f_{rosen} , the EP-CMA- m scheme cannot reach the performance of SARP, though on f_{rosen} the EP-CMA- m schemes perform as good as CMA-ES (Fig. 5).

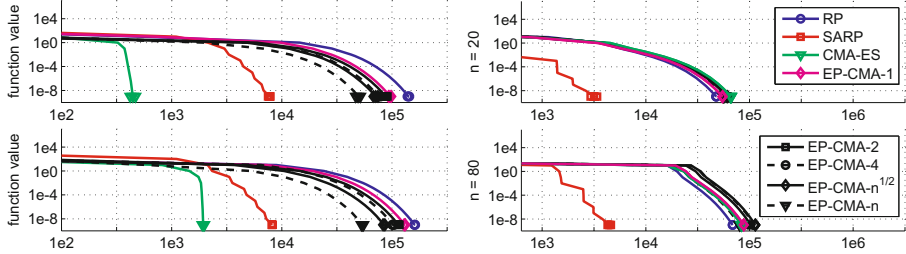


Fig. 5. Evolution of FVAL vs. #ITS on f_{two} with $L = 1\text{E}4$ (left) and f_{rosen} (right) in $n = 20$ and $n = 80$ dimensions. For 51 runs we recorded #ITS needed to reach FVAL of $1\text{E}-9$. The trajectory realizing the median values is depicted, mean and one standard deviation are indicated by markers.

CMA-ES. Fig. 4 shows nicely the quadratic dependence of the performance of CMA-ES on the dimension n , see also [26] where we report the data for all considered dimensions. The #ITS of the Random Pursuit schemes (RP, SARP) to reach the target accuracy increases only linearly (the relative performance ($\#ITS/n$) is constant over the dimensions). In the dimensions $n \leq 100$ considered here, CMA-ES is the best performing scheme on f_{exp} (Fig. 3) and f_{two} (Fig. 5); on f_{lin} the EP-CMA- m schemes match its performance for $m \geq 4$ (Fig. 3). A notable exception is the behavior on the non-convex f_{rosen} , where only SARP can accelerate and the other schemes, including CMA-ES, require over 10 times more #ITS to reach the same accuracy (Fig. 5).

4 Discussion and Conclusions

In this contribution we emphasize the importance of accelerated random gradient schemes [15, 22]. Each iteration in SARP has only linear complexity, yet the scheme takes correlations between successive iterates into account. In CMA-ES, such correlations are collected in the evolution path [18, 23] and stored in the covariance matrix. This requires $\Theta(n^2)$ simple operations per iteration. The proposed EP-CMA-1 uses as well the information of the evolution path to bias the search, but does not store a full-rank covariance matrix.

Line Search. We empirically tested two Random Pursuit algorithms with an exact line search oracle. Such an oracle is in general not available for general black-box optimization problems and the line search must for instance be implemented as bisection search (cf. [14, 27]) at the expense of additional function evaluations per iteration. The empirical data shows that both Random Pursuit algorithms do perform well if a simple adaptive step size scheme is used instead of the line search. This makes both schemes (especially SARP) promising candidates for black-box optimization, also in high dimension, as the runtime scales only linearly with the dimension. Up to our knowledge, no experimental results for SARP with adaptive step size have been published yet (the authors in [14] considered a line search with high accuracy, almost like SARP-exact).

Acceleration in EP Schemes. Our empirical results show that the sole use of the evolution path can lead to astonishingly good performance—depending on the problem and its eigenvalue spectrum. The speed-up of EP-CMA-1 on f_{lin} can be explained by the fact that the condition number of the problem drops once the algorithm has learned the most insensitive direction. Hence, the acceleration can be explained by formula (1) rather than (2). For SARP the situation is more promising. The data indicates that the convergence on f_{exp} and f_{two} is as described in (2). The same seems to be true on the non-convex f_{rosen} where SARP needs an order of magnitude less #ITS than all other schemes, including CMA-ES. Only on f_{lin} this does not hold, as SARP is only one order of magnitude faster than RP. Consider the update (4). By expansion we obtain

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \beta(\sigma_{k+1}\mathbf{u}_{k+1} + \beta(\mathbf{x}_k - \mathbf{x}_{k-1})) = \mathbf{x}_k + \sum_{i=1}^{k+1} \beta^{k+2-i} \sigma_i \mathbf{u}_i. \quad (6)$$

We see that the drift is a weighted average of the previous steps $\sigma_i \mathbf{u}_i$. The discount factor β is the expected convergence rate. Therefore, the influence of a step $\sigma_i \mathbf{u}_i$ on \mathbf{y}_{k+1} is roughly the same for all $i = 1, \dots, k$. In contrast to this, the evolution path \mathbf{p}_k stores only information of the directions of the last steps (but no step sizes). The discount factor is approximately $1 - 2/n$. Although the evolution path \mathbf{p}_k is a cumulation of all old steps, the weigh of old steps is exponentially small compared to the influence of the newest steps. We might conclude that the mechanism of accelerated random schemes like SARP is therefore inherently different to the concept of the evolution path, supporting reports in [27]. However, we cannot rule out the possibility, that with a different choice of internal parameters of EP-CMA-1 the difference to SARP could be reduced.

Limited Memory Schemes. The performance of the proposed EP-CMA- m schemes uniformly increases for larger parameters m , as well as the complexity of each single iteration. An optimal trade-off for the parameter m has to be found, depending on the dimension n and the cost of individual function evaluations. The data shows that the EP-CMA- m schemes can dramatically improve the performance of simple random search already for small values of m . The speed-up depends crucially on the eigenvalue spectra of the objective function. It seems that these schemes can not reach the performance of the related variants in [21].

We generally conclude, that the here proposed algorithmic schemes with linear iteration complexity could be a promising way to handle high dimensional black-box optimization problems. However, the empirical data suggest that there is an intrinsic limitation for the EP schemes, as they depend on the eigenvalue spectrum of the objective function. This behavior is not observed for SARP. We like to advocate that features of accelerated schemes (like SARP) should therefore be taken seriously into account when facing high dimensional problems.

References

1. Polyak, B.: Introduction to Optimization. Optimization Software - Inc. (1987)
2. Nesterov, Y.: Introductory Lectures on Convex Optimization. Kluwer (2004)

3. Broyden, C.G.: The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA J. of Appl. Math.* 6(1), 76–90 (1970)
4. Fletcher, R.: A new approach to variable metric algorithms. *The Computer Journal* 13(3), 317–322 (1970)
5. Goldfarb, D.: A Family of Variable-Metric Methods Derived by Variational Means. *Mathematics of Computation* 24(109), 23–26 (1970)
6. Nocedal, J.: Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation* 35(151), 773–782 (1980)
7. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45(1-3), 503–528 (1989)
8. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady* 27(2), 372–376 (1983)
9. Nesterov, Y.: Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming* 110(2), 245–259 (2007)
10. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization. Submitted to *SIAM Journal on Optimization* (2008)
11. Schumer, M., Steiglitz, K.: Adaptive step size random search. *IEEE Transactions on Automatic Control* 13(3), 270–276 (1968)
12. Rechenberg, I.: *Evolutionstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog (1973)
13. Mutseniyeky, V.A., Rastrigin, L.A.: Extremal control of continuous multi-parameter systems by the method of random search. *Eng.Cyb.* 1, 82–90 (1964)
14. Stich, S.U., Müller, C.L., Gärtner, B.: Optimization of convex functions with Random Pursuit. *SIAM Journal on Optimization* 23(2), 1284–1309 (2013)
15. Nesterov, Y.: *Random Gradient-Free Minimization of Convex Functions*. Technical report, ECORE (2011)
16. Leventhal, D., Lewis, A.S.: Randomized Hessian estimation and directional search. *Optimization* 60(3), 329–345 (2011)
17. Stich, S.U., Gärtner, B., Müller, C.L.: Variable Metric Random Pursuit (2012) (submitted), <http://arxiv.org/abs/1210.5114>
18. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaption in Evolution Strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
19. Hansen, N., Muller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* 11(1), 1–18 (2003)
20. Knight, J.N., Lunacek, M.: Reducing the Space-time Complexity of the CMA-ES. In: *GECCO 2007*, pp. 658–665. ACM (2007)
21. Loshchilov, I.: A Computationally Efficient Limited Memory CMA-ES for Large Scale Optimization. To appear *GECCO* (2014), <http://arxiv.org/abs/1404.5520>
22. Lee, Y.T., Sidford, A.: Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In: *FOCS*, pp. 147–156. IEEE (2013)
23. Ostermeier, A., Gawelczyk, A., Hansen, N.: Step-size adaptation based on non-local use of selection information. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) *PPSN 1994*. LNCS, vol. 866, pp. 189–198. Springer, Heidelberg (1994)
24. Igel, C., Suttorp, T., Hansen, N.: A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategies. In: *GECCO*, pp. 453–460 (2006)

25. Sun, Y., Schaul, T., Gomez, F., Schmidhuber, J.: A linear time natural evolution strategy for non-separable functions. In: Proc. 15th Genetic and Evolutionary Computation Conference Companion, pp. 61–62. ACM (2013)
26. Stich, S.U.: Supplementary Online Mat (2014), <http://arxiv.org/abs/1406.2010>
27. Stich, S.U., Müller, C.L.: On Spectral Invariance of Randomized Hessian and Covariance Matrix Adaptation Schemes. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 448–457. Springer, Heidelberg (2012)

Stopping Criteria for Multimodal Optimization

Simon Wessing¹, Mike Preuss², and Heike Trautmann²

¹ Department of Computer Science, TU Dortmund, Germany
`simon.wessing@tu-dortmund.de`

² Information Systems and Statistics Group, University of Münster, Germany
`{mike.preuss,trautmann}@uni-muenster.de`

Abstract. Multimodal optimization requires maintenance of a good search space coverage and approximation of several optima at the same time. We analyze two constitutive optimization algorithms and show that in many cases, a phase transition occurs at some point, so that either diversity collapses or optimization stagnates. But how to derive suitable stopping criteria for multimodal optimization? Experimental results indicate that an algorithm's population contains sufficient information to estimate the point in time when several performance indicators reach their optimum. Thus, stopping criteria are formulated based on summary characteristics employing objective values and mutation strength.

Keywords: Multimodal optimization, global optimization, multiobjective selection, convergence detection, stopping criteria.

1 Introduction

For quite some time, global optimization has been the predominant research direction in single-objective *evolutionary computation* (EC). While algorithms for obtaining more than one good solution at once have been investigated already in the 1970s (see [1] for a survey), the term *multimodal optimization* (MMO) has become publicly known only lately. It may be seen as superordinate concept that contains *niching* and related approaches, with the overall task to obtain a set of diverse but very good solutions. It is easy to imagine that such behavior is useful in many real-world applications, because it leaves more options to the decision maker (related arguments apply to multiobjective optimization).

In contrast to global optimization methods, MMO algorithms always employ populations and/or archives, and next to objective values, diversity is an important issue: the finally chosen best solutions should not at all be similar but be located in different search space regions. However, the interplay between reaching good objective values but keeping search diverse has not been investigated much from a general perspective, without focusing on a certain algorithm and/or optimization problem. Authors usually refer to exploitation/exploration balance, which means that there is a contradiction between improving solutions and covering the search space well. However, recent work on multiobjectivization-based selection criteria for MMO [2] suggests that it is possible to realize compromises

between the two, such that diversity is kept and still the good solutions are improved further. In §3, we show that in most cases, the balance holds only temporarily: after some time, it usually breaks and multimodal performance (dealt with in §2) degrades again. Phenomenologically, this means that the algorithm moves in direction of one extreme (see Fig. 1): either it focuses on one or few attraction basins or it emphasizes diversity so that local optimization in the separate basins becomes ineffective. This naturally calls for stopping criteria as they are, e. g., known in multiobjective optimization (see §4). We do not state that at the determined point in time, optimization shall just be cancelled. But it undergoes a phase transition after which the algorithm does not sufficiently balance both goals any more, so that it may be supplemented with other techniques as local searches. It does not seem reasonable just to continue runs.

The first goal of this paper is to document this phase transition and provide data on where it can be expected in a run for different selection types, based on a simple model algorithm that may serve as blue print for more complex methods in MMO. The second goal is to suggest (§4), experimentally assess (§5) and discuss (§6) stopping criteria that detect the right time for a behavior change of the algorithm. Differently from the situation in single- or multiobjective optimization, the important indicators cannot be observed directly in a real-world application scenario. We would have to know in advance where the different optima are located to compute the indicators. However, we can offer criteria for mutation adaptive and non-adaptive optimization algorithms that provide a good estimation of the point in time when the phase transition occurs, so that measures against a degeneration of the optimization process can be taken.

2 MMO Performance Indicators and Model Algorithms

Several different approaches exist to measure performance of multimodal optimization algorithms [3]. To precisely assess the approximation of the optima, at least their locations and objective values have to be known. This information, and above all the exact shape of corresponding attraction basins, is of course only available for benchmarking purposes. In this case, the goal in one way or another is to measure deviations from the local optima in objective and/or in decision space. After carrying out our initial investigations (see §3) for all indicators in [3], we are focusing the presentation on the quality indicator *averaged Hausdorff distance* (AHD) [4], which is a natural advancement of the well-known indicator peak distance (PD) [3,5]. It is defined as

$$\text{AHD}(\mathcal{P}, \mathcal{Q}) = \max \left\{ \frac{1}{m} \sum_{i=1}^m d_{\text{nn}}(\mathbf{z}_i, \mathcal{P}), \frac{1}{\mu} \sum_{i=1}^{\mu} d_{\text{nn}}(\mathbf{x}_i, \mathcal{Q}) \right\},$$

where $\mathcal{P} = \{\mathbf{x}_1, \dots, \mathbf{x}_\mu\}$ is the approximation set, $\mathcal{Q} = \{\mathbf{z}_1, \dots, \mathbf{z}_m\}$ is the set of optima, and $d_{\text{nn}}(\mathbf{x}, \mathcal{P}) = \min\{\|\mathbf{x} - \mathbf{y}\|_2 \mid \mathbf{y} \in \mathcal{P} \setminus \{\mathbf{x}\}\}$. PD is equivalent to the first term inside the maximum (which is also known as inverted generational distance). AHD is attractive, because it exhibits a continuous behavior over the

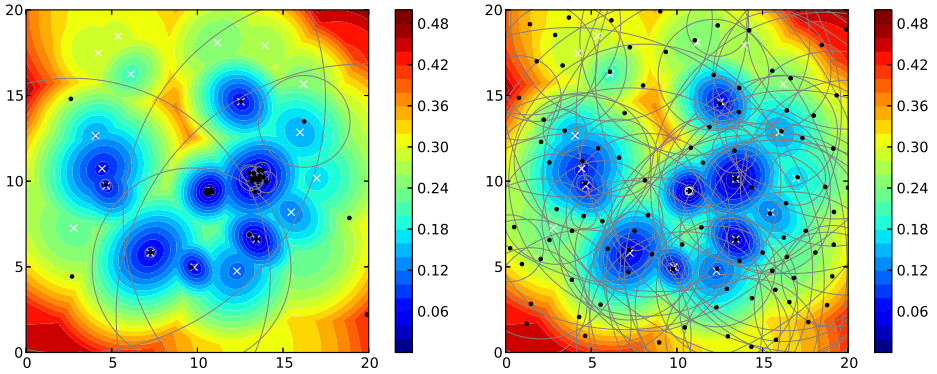


Fig. 1. Populations (black dots) after 5000 function evaluations with different selection variants (left: SV4, right: SV7). White crosses mark the local optima, a white circle the global one. Gray circles denote the mutation strengths of the respective individuals.

whole range from very bad to very good approximations. This is in contrast to, e. g., the basin ratio (BR), which measures the fraction of attraction basins that contain a point of the population. However, BR is still an informative indicator, as it has known minimal and maximal values. Other indicators, as PD or peak inaccuracy, are somewhat correlated to AHD, and thus our developments should be transferable to some extent.

Two simple evolutionary algorithms (EA) are considered in this paper. While both employ nearest-better distances $d_{\text{nb}}(\mathbf{x}, \mathcal{P}) = \min\{\|\mathbf{x} - \mathbf{y}\|_2 \mid f(\mathbf{y}) < f(\mathbf{x}) \wedge \mathbf{y} \in \mathcal{P}\}$ in their selection and use gaussian mutations for variation, they exhibit very different behaviors. The first algorithm uses a multiobjective selection with $d_{\text{nb}}(\mathbf{x}, \mathcal{P})$ as a second objective. The ranking is established by non-dominated sorting (and each non-dominated front is sorted by objective value). The second algorithm uses truncation selection on a lexicographic ordering according to the tuples $(-d_{\text{nb}}(\mathbf{x}, \mathcal{P}), f(\mathbf{x}))$. (Note that reversing the order of the criteria would essentially lead to a conventional single-objective EA.) These selections have been defined as SV4 and SV7, respectively, in [2], and we adopt these names in the following. Details and pseudocode also can be found in [2].

3 Initial Investigations

For the analysis of the algorithms' behavior, we use the following experimental setup. A budget of 10^5 objective function evaluations is allocated for optimization of the multimodal test problems described in [2] with a (100+100)-EA. The test problems are generated by taking the minimum of random samples of unimodal functions (so-called peaks, although pointing downwards). These samples can exhibit different global structures, which will be called *random*, *linear*, and *funnel* in the following. For further details we refer to [2]. For variation, we are using an isotropic mutation operator with gaussian random numbers and initial

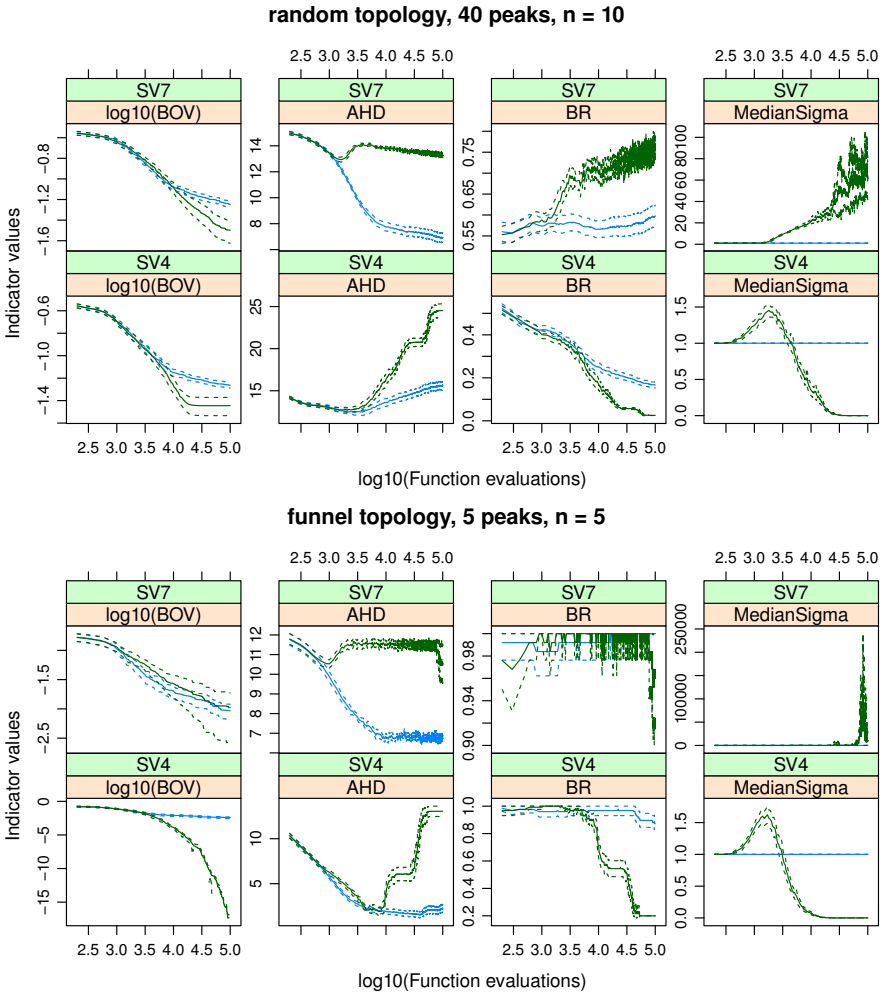


Fig. 2. EAs with fixed (blue) and self-adaptive (dark green) σ . Solid lines show mean values of 25 repeats, dashed lines are 95% pointwise confidence bands. BOV denotes best objective value and MedianSigma the median mutation strength in a population.

step size $\sigma_{\text{init}} = 1$. This operator can be used with either fixed or self-adaptive step sizes. In the latter case, the learning parameter is $\tau = 1/\sqrt{2n}$, according to recommendations of [6] for multimodal problems. Recombination is disabled.

Survivor selection is done by the two alternatives described in §2. Figure 1 shows snapshots of self-adaptive SV4- and SV7-EAs after 5000 function evaluations on a two-dimensional problem with 20 peaks and funnel topology. It can be seen that solutions close to optima possess small mutation strengths, while other solutions exhibit diverging step sizes. From Fig. 2 it is evident that SV7-EA constantly explores the search space, while SV4-EA will sooner or later converge

to a single optimum. The figure shows the average performance on two selected problem classes over time. For SV4, step size adaptation first leads to a slight increase of σ before the conventional convergence to zero starts. Therefore, the self-adaptive SV4-EA is the best suited for global optimization among the tested algorithms, as it does converge to one single optimum at some point, but does this later than a conventional single-objective EA. Note that SV4-EA has the same structure as [7], but is expected to yield better global optimization performance due to step-size adaptation and use of d_{nb} . In this case, also the existing stopping criteria for single-objective optimization (see §4) can be applied.

SV7, on the other hand, exhibits a permanent tendency towards larger σ , which is sometimes beneficial but often leads to a deterioration of quality in the late stages of the run. Additionally, the algorithms' performance also depends on problem features as the search space dimension and the number of local optima. Here, low-dimensional, weakly multimodal problems favor SV4, while SV7 seems more adequate in the opposite case. Thus, if a diverse set of good solutions is sought as a result, special stopping criteria for multimodal optimization should be employed in any case.

4 Stopping Criteria

So far, research on stopping criteria within the field of EAs concentrated on assessing the convergence behavior of the respective algorithms. Formal analysis of convergence behavior is difficult and only possible for special and usually simplified cases. As optimality criteria such as the Karush-Kuhn-Tucker conditions usually cannot be applied in the black-box scenario due to the lack of sufficient gradient information, heuristic approaches were introduced to check whether the expected improvement in convergence is worth the additional amount of function evaluations which has to be spent. So far, to the best of our knowledge, no specific stopping criteria for multimodal optimization have been introduced, which have to be designed to focus on tracking the trade-off between maintaining diversity and ensuring sufficient convergence.

An overview about recent approaches for multiobjective optimization is provided in [8]. As most of the methods rely on analyzing (single-objective) performance indicators, the approaches in principle could be transferred to single-objective optimization tasks as well. However, none of these criteria allows for adequately terminating an EA in the multimodal situation in which the population is desired to converge while simultaneously maintaining diversity. In single-objective optimization the same problem exists. In [9] existing termination conditions in the single-objective case are discussed which consist of either theoretically motivated approaches [10,11], movement criteria [12], or qualified runtime distributions [13]. To our knowledge, criteria based on statistical hypothesis testing are surprisingly uncommon. It shall also be stated that contrary to intuition, deriving criteria for single-objective algorithms is *not* necessarily simpler than for multiobjective ones. As, e. g., the list of criteria in [14] demonstrates, there is much more information available for the latter case, so that many

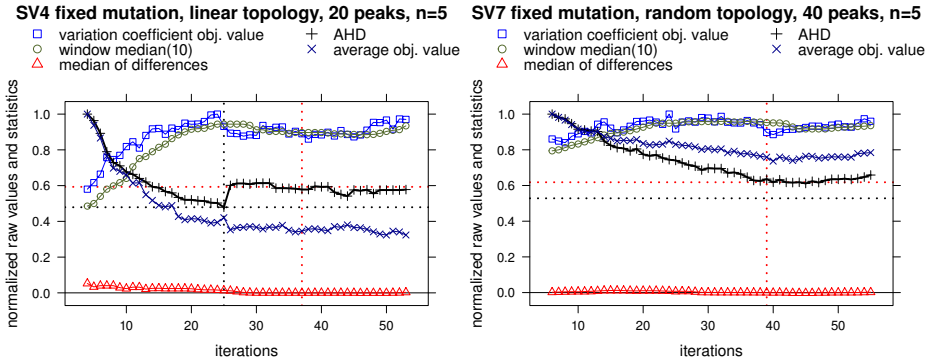


Fig. 3. Exemplary runs of SV4- and SV7-EAs with fixed σ . In both cases, the variation coefficient of the objective value is a suitable signal for stopping. The optimal stopping point is marked with a black dotted line, the actual stopping point with a red one.

more generic criteria may be established. Algorithm internal criteria, as, e. g., integrated into CMA-ES, focus on a concentrated population and the convergence to a single optimum, which is reflected by a fading mutation strength. Thus, the challenges of multimodal optimization are not properly reflected within the existing criteria which focus on stagnation related to desired convergence.

Stopping Criteria for Multimodal Optimization (SMMO): In contrast to the requirements for single- and multiobjective stopping criteria, we cannot directly observe the measures we are actually interested in (there: best objective value and hypervolume, respectively). In order to find a signal that can be exploited as stopping criterion due to its correlation to the course of the AHD indicator, we have investigated a large number of time series by visual inspection, e. g., the best and average objective values, the standard deviation of the objective values, the average mutation strength, its standard deviation, the *coefficients of variation* (standard deviation divided by mean value, CV) of objective values and mutation strengths, and diversity indicators [3].

We found two signals that appear to be useful. For the self-adaptive case, the mutation strength on average (Fig. 2) experiences a peak when the AHD reaches its minimum. In most cases, this behavior can also be found when looking at individual runs. A slightly less obvious correspondence that may be used for the fixed mutation strength algorithms exists between the CV of the objective values and the AHD. In many cases, the CV starts to decline when the AHD passes its minimum, as displayed in Fig. 3. As the raw signal shows fluctuations in both cases, we employ the window median $\tilde{x}_w(t) = \text{median}(x_{(t-w+1):t})$, where x_t is a time series and t runs from 1 to t_{\max} , over the median mutation strength and the CV of the objective values, respectively. After some initial experimentation, we chose $w = 10$ as window size. From that, we compute the window median $\tilde{x}'_w(t)$ of the forward differences of $\tilde{x}_w(t)$ in order to find the point in time when the original value is decreasing considerably. We stop as soon as the median of

Table 1. Factors for the experiment in §5

Factor	Type	Symbol	Levels
Problem topologies	environmental		{random, linear, funnel}
Number of variables	environmental	n	{2, 3, 5, 10}
Number of peaks	environmental	N	{5, 20, 40}
Selection variants	control		{SV4, SV7}
Mutation strength	control		{fixed, self-adaptive}

Table 2. Differences between the optimal (w.r.t. AHD) stop generation and the suggested one as well as the percentages of generations after *StopGen* with higher AHD

		StopGenAHD – StopGen			% higher AHD after			
	Strategy	Criterion	LQ	Median	UQ	LQ	Median	UQ
SV4	SA	MutStrength	–6	1	21	97.1	99.4	99.9
	NonSA	VarCoeffObj	–38	–11	95	67.9	95.1	98.9
SV7	SA	MutStrength	–57	351	680.5	14.5	35.1	61.2
	NonSA	VarCoeffObj	304.8	568	784.2	0.9	12.1	41.2

differences, $\tilde{x}'_w(t) = \tilde{x}_w(\tilde{x}_w(t) - \tilde{x}_w(t-1))$, gets negative for the first time (the first w time steps are ignored).

5 Experimental Evaluation of SMMO

Research Question: Do the stopping criteria of §4 provide a reasonable performance?

Pre-experimental Planning: The stopping criteria in §4 were selected after a first visual inspection of several summary characteristics. After some preliminary investigations, we decided to test the CV-based criterion only with fixed σ as the mutation strength criterion seemed superior (when available).

Task: The task of the stopping criteria is to abort the runs early with as few loss of performance as possible. The key criterion for us is the AHD indicator.

Setup: The bulk of the setup was already described in §3. Table 1 summarizes all the factors for this full-factorial experiment. For each configuration, five random test instances are drawn and five independent algorithm runs are carried out per problem instance, leading to a total of 25 repeats per configuration.

Results: Figures 4 and 5 show how much worse the obtained AHD values are for early stopping in comparison to the best value of the same algorithm run that would be obtained sometime during the full 10^5 function evaluations. Table 2 contains another investigation of the same data, focusing on the differences of the actual stop generation and the respective one with minimum AHD value. Furthermore, the percentage of generations after the stop generation where the obtained AHD value was higher than the one in the stop generation is provided.

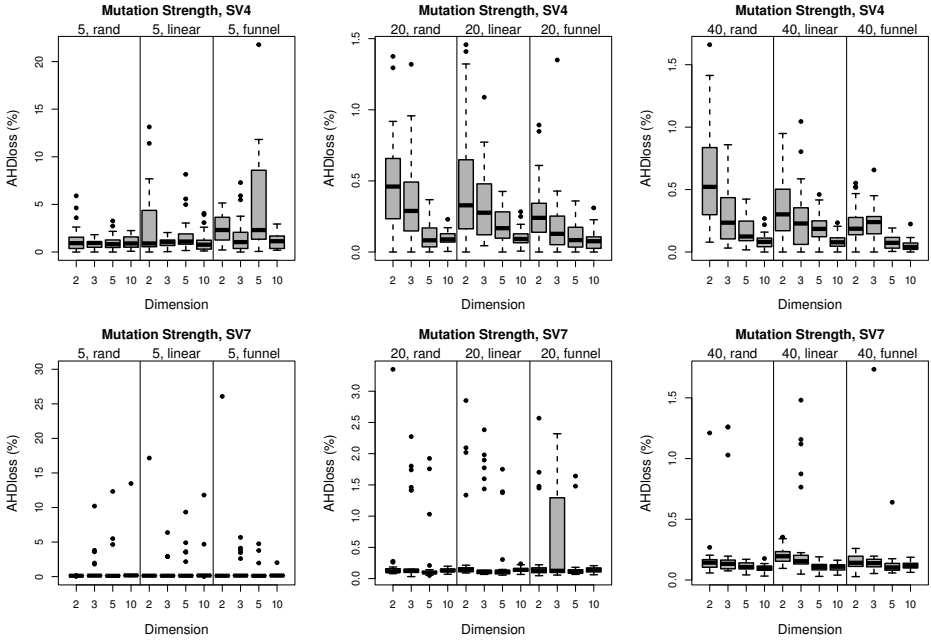


Fig. 4. Performance losses through the stopping criterion based on σ

Observations: For SV4, the loss of AHD performance is decreasing with increasing n if N is 20 or 40. SV7 shows the opposite behavior, especially with the CV-based criterion. If the problem contains only five peaks, the AHD loss is generally higher, especially with the σ -criterion (although the absolute values may still be better than with the CV). Table 2 reflects that the recommended stop generation does not differ much from the respective one with minimum AHD for SV4. Moreover, an almost neglectable percentage of obtained AHD values after stopping results in smaller AHD. SV7 shows a different behavior, the interquartile ranges of both statistics are relatively large and the median levels differ quite much from the respective ones of SV4.

Discussion: The seemingly worse performance with five peaks may occur because these problems are relatively easy and therefore the obtained AHD values are close to zero. So, even small absolute deviations appear as high relative deviations. On SV7 the losses are smaller, which is probably because the AHD values are generally fluctuating less. For SV7, the statistics in Table 2 reflect that the AHD quality usually does not show an obvious decreasing tendency after StopGenAHD but rather a fluctuating behavior around the minimum AHD. Applied to SV4, the suggested stopping criteria successfully detect an adequate stopping generation in the vicinity of StopGenAHD.

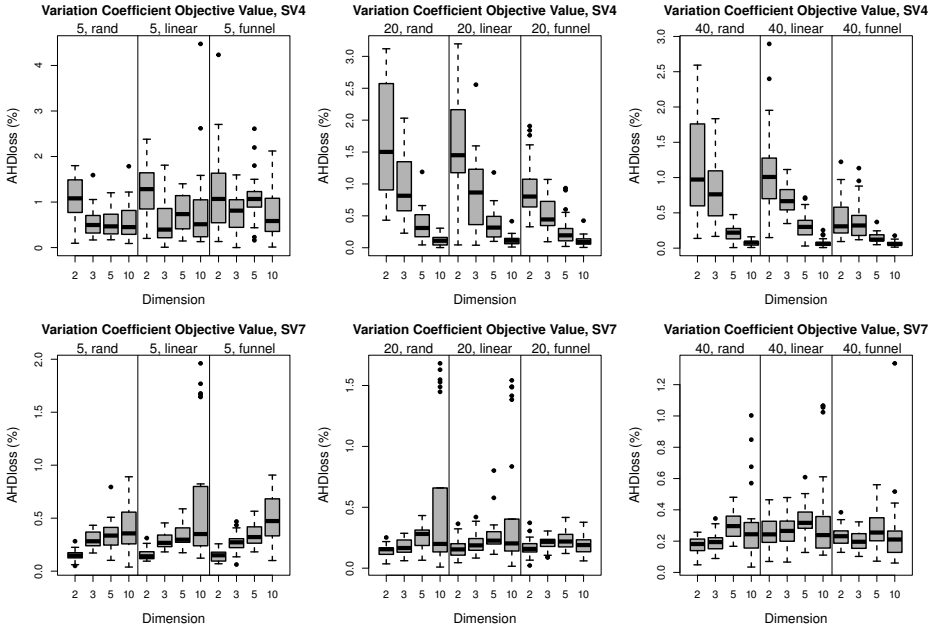


Fig. 5. Performance losses AHD through the criterion based on the variation coefficient

6 Conclusions and Outlook

By means of systematic experiments we are able to show that transition phases between maintaining diversity and converging to single optima exist. While this is intuitive for classical evolution strategies, this effect also can be observed for strategies which explicitly address multimodality as SV4. Structural differences compared to SV7 are present, for which a large percentage of local optima are successfully approximated during the whole algorithm run due to extensive exploration of the search space.

Decreasing AHD between the set of local optima and the population coincides with increasing approximation quality in the multimodal setting. Indicators based on the mutation strength (self-adaptive strategies) or the variation coefficient of objective values (fixed step sizes) could be set up which appropriately reflect the AHD behavior over time which is naturally unknown within the actual algorithm run. The suggested stopping criteria, in most cases, recommended stopping generations which simultaneously ensure the coverage of the modes as well as sufficient proximity to the latter. However, they face greater challenges for decreasing number of modes but improve for increasing search space dimensionality for the mutation strength criterion.

In future work, we will explicitly analyze the influence of self-adaption of the mutation strength on algorithm performance. Moreover, the suggested stopping criteria will be included into more sophisticated MMO algorithms.

Acknowledgments. Heike Trautmann and Mike Preuss acknowledge support by the European Research Center for Information Systems (ERCIS).

References

1. Das, S., Maity, S., Qu, B.Y., Suganthan, P.N.: Real-parameter evolutionary multimodal optimization – a survey of the state-of-the-art. *Swarm and Evolutionary Computation* 1(2), 71–88 (2011)
2. Wessing, S., Preuss, M., Rudolph, G.: Niching by multiobjectivization with neighbor information: Trade-offs and benefits. In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 103–110 (2013)
3. Preuss, M., Wessing, S.: Measuring multimodal optimization solution sets with a view to multiobjective techniques. In: Emmerich, M., Deutz, A., Schütze, O., Bäck, T., Tantar, E., Tantar, A.A., Moral, P.D., Legrand, P., Bouvry, P., Coello, C.A. (eds.) *EVOLVE – A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*. AISC, vol. 227, pp. 123–137. Springer, Heidelberg (2013)
4. Schütze, O., Esquivel, X., Lara, A., Coello Coello, C.A.: Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 16(4), 504–522 (2012)
5. Stoean, C., Preuss, M., Stoean, R., Dumitrescu, D.: Multimodal optimization by means of a topological species conservation algorithm. *IEEE Transactions on Evolutionary Computation* 14(6), 842–864 (2010)
6. Beyer, H.G., Schwefel, H.P.: Evolution strategies – a comprehensive introduction. *Natural Computing* 1(1), 3–52 (2002)
7. Tran, T.D., Brockhoff, D., Derbel, B.: Multiobjectivization with NSGA-II on the noiseless BBOB testbed. In: *Proceeding of the Fifteenth Annual Conference companion on Genetic and Evolutionary Computation Conference Companion, GECCO 2013 Companion*, pp. 1217–1224. ACM (2013)
8. Wagner, T., Trautmann, H., Martí, L.: A taxonomy of online stopping criteria for multi-objective evolutionary algorithms. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) *EMO 2011*. LNCS, vol. 6576, pp. 16–30. Springer, Heidelberg (2011)
9. Trautmann, H., Wagner, T., Naujoks, B., Preuss, M., Mehnen, J.: Statistical methods for convergence detection of multi-objective evolutionary algorithms. *Evolutionary Computation Journal* 17(4), 493–509 (2009)
10. Hernandez, G., Wilder, K., Nino, F., Garcia, J.: Towards a self-stopping evolutionary algorithm using coupling from the past. In: *GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pp. 615–620. ACM (2005)
11. Safe, M., Carballido, J.A., Ponzoni, I., Brignole, N.B.: On stopping criteria for genetic algorithms. In: Bazzan, A.L.C., Labidi, S. (eds.) *SBIA 2004*. LNCS (LNAI), vol. 3171, pp. 405–413. Springer, Heidelberg (2004)
12. Zielinski, K., Laur, R.: Stopping criteria for a constrained single-objective particle swarm optimization algorithm. *Informatica* 31(1), 51–59 (2007)
13. Hoos, H.H., Stützle, T.: *Stochastic Local Search – Foundations and Applications*. Morgan Kaufmann, San Francisco (2004)
14. Sastry, K.: Single and multiobjective genetic algorithm toolbox for Matlab in C++. Technical Report 2007017, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign (2007)

VLR: A Memory-Based Optimization Heuristic

Hansang Yun, Myoung Hoon Ha, and Robert Ian McKay

School of Computer Science and Engineering, Seoul National University
Seoul, 151-744, Republic of Korea

Abstract. We suggest a novel memory-based metaheuristic optimization algorithm, VLR, which uses a list of already-visited areas to more effectively search for an optimal solution. We chose the Max-cut problem to test its optimization performance, comparing it with state-of-the-art methods. VLR dominates the previous best-performing heuristics. We also undertake preliminary analysis of the algorithm's parameter space, noting that a larger memory improves performance. VLR was designed as a general-purpose optimization algorithm, so its performance on other problems will be investigated in future.

Keywords: Optimization, Metaheuristics, Max-cut problem, memory.

1 Introduction

We introduce a novel metaheuristic optimization algorithm which uses a list of already-visited areas (the Visited-Local-Region – VLR) to improve the efficiency of exploration. VLR extends a general local search approach, guiding the system to avoid regions which are unlikely to have favorable solutions or near which we have already searched. The VLR technique is inspired by the biological mechanisms of microRNA, small non-coding RNA molecules which regulates gene expression [1]. The search step size is controlled by a “threshold of uncertainty” (TU), which resembles the temperature in Simulated Annealing (SA). Unlike SA's temperature, TU does not decrease monotonically, but depends on the VLR – we describe it in detail later. We apply VLR to the Maximum Cut (Max-cut) problem with good results.

The remainder of this paper is organized as follows. In Section 2, we introduce previous work on the Max-cut problem. Section 3 describes the VLR heuristic and its application. Section 4 presents the experiment settings for Max-cut, with the computational results being presented in Section 5. We draw conclusions and point out future directions in the last section.

2 Background

2.1 Search Methods

We propose a stochastic local search algorithm and evaluate it on the Max-cut problem, though it is general in form, and may be useful for other problems. Most

stochastic search methods do not employ an explicit long-term memory, either foregoing memory altogether (stochastic hillclimbing and its variants) or relying on an implicit memory either in the population (genetic and swarm algorithms) or in a probability structure (estimation of distribution and ant algorithms). Only a few algorithms, of which tabu search [2] is the most prominent, explicitly remember details of the search space. The algorithm we propose here resembles tabu search in retaining an explicit memory of areas in the search space that it has previously visited. However tabu search uses the memory to determine its neighborhood structure; this algorithm differs in many details, most obviously in using the memory to affect the acceptance criterion rather than the neighborhood structure. It will be described in more detail in the next section.

2.2 The Max-Cut Problem

Let V be a set of n vertices, E a set of edges (i, j) with $i, j \in V$, and W a set of weights w_{ij} on the edges $(i, j) \in E$. For a graph $G = (V, E)$, the Max-cut problem seeks a cut (S, S^c) that maximizes the sum of the weights on the edges between S and its complement S^c :

$$\text{maxcut}(G) = \max_{S \subseteq V} \left(\sum_{u \in S, v \in S^c} w_{uv} \right)$$

In spite of its simple conceptualization, this problem is remarkable for its practical applications such as the design of VLSI circuits [3,4], and the determination of ground states of spin-glass models in statistical physics [5]. Since Karp [6] proved it NP-Complete, various algorithms have been used to achieve better solutions with limited computing resources.

Most recent studies focused on heuristic techniques. Burer et al. [7] proposed a rank-2 relaxation heuristic, CirCut, that achieves better solutions than any previous, and can handle bigger problems in shorter computational time. Festa et al. [8] combined a greedy randomized adaptive search procedure (GRASP), variable neighborhood search (VNS), and path relinking (PR) to form a hybrid randomized method. Their VNSPR produces high quality solutions, but with high computational effort (time). These methods can be further hybridized with others, such as CirCut or the Goemans and Williamson algorithm.

The subsequent Scatter Search (SS) heuristic of Martí et al. [9] obtained better performance than CirCut. Kochenberger et al. [10] applied a new Tabu Search algorithm, Diversification-Driven Tabu Search (DDTS) [11], demonstrating its solution quality and computational efficiency on Max-cut problems up to 10,000 vertices. Song and Li's [12] HMA combines a memetic algorithm (MA) with semidefinite programming for initialization, as in [13] GW+Random. It creates a better initial population leading to a higher score for Max-cut than the SS in [9] over many different problems.

We compared our approach with state-of-the-art heuristics, namely SS, HMA, DDTS. Our new method out-performs these older heuristics in most cases.

3 Method

Visited-Local-Region (VLR) is a general-purpose metaheuristic optimizing a target function $f(x)$ over solution vectors x . In the Max-cut problem, x is a binary vector representing which elements belong to set S , and $f(x)$ denotes the value sum for the cut(S, S^c).

We begin with a simpler version, Visited-Local-Hill (VLH), with two main characteristics. It memorizes already-visited local hills, represented by their local optima. Each hill has an attribute, the escape count (EC), proportional to the number of visits (with user-set constant of proportionality d). It controls the extent of exploration by a “threshold of uncertainty” τ , differing from SA’s temperature in two ways:

1. Its value is reversed (low values mean more exploration).
2. It can both increase and decrease during search.

We give the explanation in three parts: exploration control in the Uncertain-Climbing method, the VLH heuristic, and the VLR extension of VLH. To simplify the explanation, we limit it to Boolean domains.

3.1 Exploration

UncertainClimbing (Algorithm 1) iteratively seeks a local optimum x . To permit crossing between hills, we allow it to climb down (the probability varying over time), unlike the random exploration that traditional stochastic hill climbing undertakes when it is stuck in a local optimum. Let x' be a bit-flip neighbor of x . We define $h(x)$ and $p(x \rightarrow x')$ as:

$$h(x) = \max(f(x) - \tau, 0)$$

$$p(x \rightarrow x') = \begin{cases} h(x')/(h(x) + h(x')) & , \text{ if } (h(x) + h(x')) > 0 \\ 0 & , \text{ otherwise} \end{cases}$$

Algorithm 1. Pseudocode of UncertainClimbing

```

01: procedure UncertainClimbing( $x, \tau$ )
02:   repeat
03:     for  $i = 1$  to  $N$  do //  $N$  is length of  $x$ 
04:       let  $x'$  be a neighbor of  $x$  obtained by flipping the  $i$ 'th bit of  $x$ 
05:       pick a random real  $r$  between 0 and 1
06:        $h(x) = \max(f(x) - \tau, 0)$ ,  $h(x') = \max(f(x') - \tau, 0)$ 
07:       if  $h(x) + h(x') > 0$  and  $r < h(x')/(h(x) + h(x'))$  then
08:         replace  $x$  with  $x'$ 
09:       end if
10:     end for
11:     increase  $\tau$ 
12:   until ( $x$  is a local optimum)
13:   return ( $x, \tau$ )
14: end procedure

```

Algorithm 2. Pseudocode of VLHmain

```

01: procedure VLHmain( )
02:   initialize  $x$  with a random solution and  $\tau$  with  $f(x)$  .
03:   repeat
04:      $(x, \tau) = \text{UncertainClimbing}(x, \tau)$ 
05:     if  $x \notin \text{VLH.keys}$  then enroll  $x$  in  $\text{VLH}$  end if
06:      $\text{VLH}[x].ec = \text{VLH}[x].ec + d$ 
07:      $\tau = \tau - \text{VLH}[x].ec$ 
08:   until (end condition)
09:   return the best found solution
10: end procedure

```

$p(x \rightarrow x')$ is the probability of accepting x' as the next solution: solutions better than x have higher rates, but worse solutions have some chance of acceptance. The propensity for exploration is managed by τ . For $f(x') \leq \tau$, $p(x \rightarrow x') = 0$, so x' is rejected; for $f(x') > \tau$, $p(x \rightarrow x') > 0$ and x' may be accepted. Until the process finds the local optimum, τ increases, decreasing the acceptance rate and making the process more eager, so that it eventually reaches the local optimum.

3.2 Visited-Local-Hill (VLH)

RNA silencing works like this: when a messenger RNA contains a genetically vulnerable feature – coding say for diabetes or cancer – microRNAs evolve to silence them by inactivating the matching sequence. VLH acts in a similar way, changing τ to move the search away from previously matched features.

VLH lists previously visited local optima, implemented as a TRIE structure. This gives match determination time dependent on the search space dimension, but not the list size. When we find a new local optimum not in the VLH list, we add it and initialize the escape count (EC) to a constant d . EC is used to decrement τ , determining the acceptance rate for exploration, and thus the distance to the next target. If EC is small relative to the basin of attraction of the local optimum, the process may revisit the same peak, in which case, we increase EC until it is large enough to exit the basin.

Overall, τ works as follows. When the process is seeking a local optimum, τ increases; when it reaches one, τ suffers a large reduction (by EC) and escapes the basin of attraction. Thus the system can act as both a local optimization method and a global one, with τ and EC adapting the algorithm to the scale of the fitness landscape. Algorithm 2 provides the full details.

3.3 Visited-Local-Region(VLR)

VLH is simple and performs competitively. However a small extension can improve it. Searching and escaping from each hill can limit the search scope. We broaden it by defining a local region embracing a few minor hills, represented by the fittest solution, y , detected in the region. During this phase, τ increases – eventually, it will exceed the maximum available $f(x)$ in the region, so search stalls; τ suffers a reduction by EC and x can move again. VLR is described in detail in Algorithms 3 and 4.

Algorithm 3. Pseudocode of VLRmain

```

01: procedure VLRmain( )
02:   initialize  $x$  with a random solution and  $\tau$  with  $f(x)$  .
03:   repeat
04:      $(x, y, \tau) = \text{UncertainClimbing2}(x, \tau)$ 
05:     if  $y \notin \text{VLR.keys}$  then enroll  $y$  in  $\text{VLR}$  end if
06:      $\text{VLR}[y].ec = \text{VLR}[y].ec + d$ 
07:      $\tau = \tau - \text{VLR}[y].ec$ 
08:   until (end condition)
09:   return the best found solution
10: end procedure

```

Algorithm 4. Pseudocode of UncertainClimbing2

```

01: procedure UncertainClimbing2( $x, \tau$ )
02:    $y = x$ 
03:   repeat
04:     for  $i = 1$  to  $N$  do //  $N$  is length of  $x$ 
05:       let  $x'$  be a neighbor of  $x$  obtained by flipping the  $i$ 'th bit of  $x$ 
06:       pick a random real  $r$  between 0 and 1
07:        $h(x) = \max(f(x) - \tau, 0)$ ,  $h(x') = \max(f(x') - \tau, 0)$ 
08:       if  $h(x) + h(x') > 0$  and  $r < h(x') / (h(x) + h(x'))$  then
09:         replace  $x$  with  $x'$ 
10:         if  $f(x) > f(y)$  then  $y = x$  end if
11:       end if
12:     end for
13:     increase  $\tau$ 
14:   until ( $x$  has been stuck ) // i.e.  $x$  is a local optimum and  $\tau \geq f(x)$ 
15:   return ( $x, y, \tau$ )
16: end procedure

```

4 Experiments

We compared VLR with three leading metaheuristics from section 2: SS [9], HMA [12], and DDTS [10]. SS is the best known algorithm for Max-cut, outperforming well-known methods such as VNSPR [8] and CirCut [7]. HMA has better performance than SS in most cases, but does not dominate. DDTS has even better performance on the same instances, and has been tested on larger problems (3000 to 10000), beyond the range on which SS and HMA were tested. All had been tested on the Gset test set of Helmberg and Rendl [14] of 54 problems, varying from 800 to 3000 vertices, so we used Gset for comparison.

The HMA tests used 30 min. on an Intel Core i5-750 2.67GHz. To calibrate with our Intel Core i7-870 2.93GHz system, we used SPEC2006 benchmarks, giving us a time budget of 27 min. We were able to compare also with SS because [12] provided timings for the same problems under both HMA and SS. [10] does not provide timing details for DDTS, so it is difficult to determine the fairness of the comparison, but we nevertheless provide the comparative attainments of the algorithms.

Ideally, such comparisons should be tested for statistical significance. This was not possible, because the per-run data for previous systems were not available, and the number of runs were too small for statistical stability. To support such comparisons in future, all our tests used 50 runs, and the raw data are available at

<http://hdl.handle.net/10371/91260>. Suitable parameter settings were determined by some more detailed exploration of VLR’s parameter space, which we detail below.

To test the importance of the region search in VLR, we compared VLH and VLR performance on G16, G21, G32, G37, and G52 from Gset. We tested the impact of list size by comparing two extreme cases, VLR lists of sizes 1 and 8,000, on G37. Finally, we tested the sensitivity of VLR’s performance to the value of the user-set parameter d , trying an exponentially increasing series of sizes $-\sqrt{|V|}/4, \sqrt{|V|}/2, \sqrt{|V|}, 2\sqrt{|V|}, 4\sqrt{|V|}$ – on G16, G21, G37, and G52.

Detailed parameter settings for the experiments are shown in Table 1.

Table 1. Experimental Parameter Settings

Parameter	Value	Parameter	value
d	$\sqrt{ V }$	the amount of τ increment	1
V	Problem dependant	Number of Runs	50
list size	$\frac{1 \text{ GByte}}{(\text{size of 1list item}) \times V }$		

5 Result and Discussion

Table 2 shows the results of comparisons between VLR and earlier methods. We show the best known record to date, together with the available data from [12] and [10]. For SS and HMA, we show their best and average performance from 10 runs. Curiously for a stochastic algorithm, the results for DDTS in [10] appear to be from single runs, so that is all we can present. For VLR, we show the best, average and median values from 50 runs, and the success rate.¹

Bold values indicate best-known performance on a problem. The values enclosed in square brackets are new best-known records. Comparing VLR with SS, VLR has superior performance on 37 problems and worse on one. Comparing with HMA, VLR wins on 30 problems and loses on one. Thus VLR almost dominates SS and HMA; although we cannot statistically test the comparison, it seems that VLR is overall a better performer. The comparison with DDTS is difficult because of the single runs for DDTS, but VLR wins on 27 problems and DDTS on 3. Overall VLR performed well, finding 20 new best-known solutions, while failing to find a known-best solution on only 3 problems. Overall, VLR’s performance on Max-cut is clearly of a high order.

Table 3 compares the performance of VLH and VLR. The result of one sample t-test whether all pairwise difference differ from 0 suggests that the region structure brings performance gains on average, though differences are fairly small.

¹ For skewed data such as typically arises in optimization, the median is generally more informative than the mean.

Table 2. Comparative Results on Gset Instances

ID	# of Verts	Best Known	SS Best	SS Avg.	HMA Best	HMA Avg.	DDTS	VLR Best	VLR Avg.	VLR Median	VLR Succ. Rate	
Number of Runs		10					1	50				
G1	800	11624	11624	11624.0	11624	11624	11624	11624	11622.9	11624	47	
G2	800	11620	11620	11620.0	11620	11620	11620	11620	11620.0	11620	50	
G3	800	11622	11622	11619.5	11622	11622	11620	11622	11622.0	11622	50	
G4	800	11646	11646	11638.5	11646	11646	11646	11646	11646.0	11646	50	
G5	800	11631	11631	11630.4	11631	11631	11631	11631	11631.0	11631	50	
G6	800	2178	2178	2174.1	2178	2178	2178	2178	2178.0	2178	50	
G7	800	2006	1996	1988.9	2006	2006	2006	2006	2006.0	2006	50	
G8	800	2005	1996	1994.7	2005	2005	2005	2005	2005.0	2005	50	
G9	800	2054	2054	2051.2	2054	2053	2054	2054	2054.0	2054	50	
G10	800	2000	2000	1999.1	2000	1999.1	2000	2000	1999.5	2000	44	
G11	800	564	564	563.8	558	558	564	564	564.0	564	50	
G12	800	556	554	550.6	556	552	556	556	556.0	556	50	
G13	800	580	578	575.8	578	578	580	[582]	582.0	582	50	
G14	800	3063	3063	3060.8	3060	3058.1	3061	3063	3061.3	3061	1	
G15	800	3050	3040	3036.8	3049	3048.8	3050	3050	3049.1	3049	7	
G16	800	3052	3044	3043.7	3050	3048.8	3052	3052	3051.0	3051	10	
G17	800	3046	3040	3038.4	3045	3043.6	3046	[3047]	3045.5	3046	1	
G18	800	991	991	985.8	989	986.9	991	[992]	991.3	991	19	
G19	800	906	905	898.9	906	904.1	904	906	905.2	906	31	
G20	800	941	941	941.0	941	941	941	941	941.0	941	50	
G21	800	931	931	931.0	931	930.9	931	931	930.8	931	47	
G22	2000	13359	13349	13314.8	13358	13349.8	13359	13359	13345.8	13358	9	
G23	2000	13342	13323	13312.6	13337	13329.3	13342	[13344]	13335.8	13337	9	
G24	2000	13337	13318	13307.8	13330	13321.8	13337	13337	13322.9	13324	6	
G25	2000	13332	13320	13313.8	13330	13322	13332	[13335]	13325.4	13328	4	
G26	2000	13328	13308	13299.7	13323	13310	13328	13326	13315.8	13316	8	
G27	2000	3336	3332	3312.0	3334	3325.8	3336	[3341]	3326.8	3330	4	
G28	2000	3295	3275	3264.9	3294	3286.9	3295	[3298]	3291.0	3294	11	
G29	2000	3404	3385	3376.0	3404	3386.5	3391	[3405]	3385.8	3386	4	
G30	2000	3407	3395	3384.5	3407	3402.8	3403	[3412]	3402.5	3403	15	
G31	2000	3305	3275	3265.8	3305	3296.3	3288	[3310]	3299.6	3301	2	
G32	2000	1406	1400	1393.2	1396	1392.2	1406	[1410]	1404.4	1404	1	
G33	2000	1378	1364	1359.4	1372	1368.4	1378	1378	1374.5	1374	3	
G34	2000	1378	1368	1361.6	1378	1375	1378	[1382]	1380.1	1380	15	
G35	2000	7680	7654	7648.5	7680	7673	7678	[7683]	7678.9	7679	2	
G36	2000	7670	7667	7654.5	7670	7665.7	7670	[7675]	7671.0	7671	1	
G37	2000	7682	7667	7660.3	7682	7674.6	7682	[7687]	7685.4	7686	9	
G38	2000	7683	7668	7659.8	7678	7669.9	7683	[7687]	7684.2	7685	3	
G39	2000	2406	2395	2388.0	2406	2396.9	2397	[2408]	2399.6	2399	9	
G40	2000	2393	2380	2375.9	2393	2389.2	2390	[2399]	2390.7	2394	1	
G41	2000	2405	2391	2385.7	2405	2401.7	2400	2405	2398.0	2405	26	
G42	2000	2478	2462	2458.1	2478	2469.1	2469	[2480]	2471.0	2472	2	
G43	1000	6660	6660	6656.2	6660	6658.7	6660	6660	6659.1	6660	48	
G44	1000	6650	6650	6648.8	6650	6649.7	6639	6650	66467.0	6650	30	
G45	1000	6654	6646	6643.0	6654	6650.1	6652	6654	6648.7	6650	16	
G46	1000	6649	6647	6640.4	6649	6645.8	6649	6649	6645.4	6648	18	
G47	1000	6665	6655	6652.9	6656	6655.2	6665	6657	6651.3	6650	9	
G48	3000	6000	-	-	6000	6000	6000	6000	6000.0	6000	50	
G49	3000	6000	-	-	6000	6000	6000	6000	6000.0	6000	50	
G50	3000	5880	-	-	5880	5880	5880	5876	5859.6	5874	15	
G51	1000	3847	3843	3839.4	3847	3843.9	3847	3847	3846.0	3846	9	
G52	1000	3849	3841	3836.1	3848	3844.8	3849	[3851]	3849.1	3849	3	
G53	1000	3849	3845	3844.3	3849	3844.9	3848	3849	3846.7	3846	2	
G54	1000	3851	3849	3846.0	3845	3842.5	3851	3851	3850.1	3850	4	

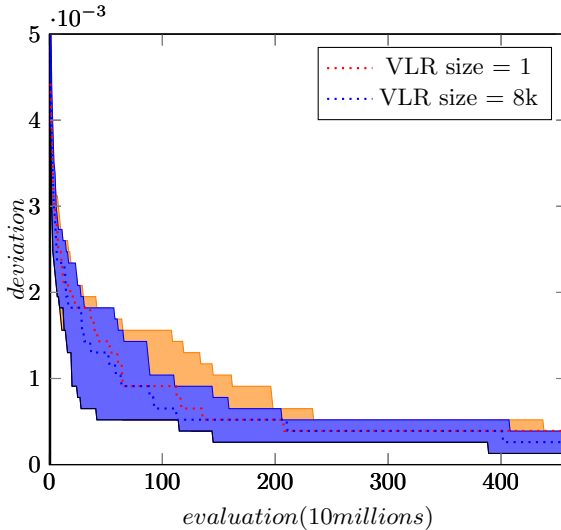
Table 3. VLH-VLR Performance Comparison

	VLH Best	VLH Avg.	VLR Best	VLR Avg.	t	p value	mean
G16	3052	3050.9	3052	3051.06	8.593	<2.2e-16	0.16
G21	931	930.76	931	930.94	11.6283	<2.2e-16	0.18
G32	1408	1404.88	1410	1404.84	-0.6851	0.4934	-0.04
G37	7688	7683.78	7687	7684.7	12.8641	<2.2e-16	0.92
G52	3850	3848.74	3851	3849.08	16.462	<2.2e-16	0.34

Table 4. Performance Comparison for VLR Parameter d

	$\sqrt{ V }/4$	$\sqrt{ V }/2$	$\sqrt{ V }$	$2\sqrt{ V }$	$4\sqrt{ V }$	t	p-value	mean
G16	3049.1	3051.58	3051.06	3048.6	3045.22	175.1456	<2.2e-16	6.36
G21	921.46	929.26	930.94	931	930.32	64.8215	<2.2e-16	9.54
G32	1392.12	1397.6	1404.84	1400.64	1393.16	131.3383	<2.2e-16	12.72
G37	7677.9	7682.62	7684.7	7678.88	7665.5	218.2509	<2.2e-16	19.2
G52	3847.04	3850	3849.08	3844.88	3840.76	236.2583	<2.2e-16	9.24

Table 4 shows the sensitivity analysis for user-set parameter d . We can see that the performance is fairly sensitive to d from the result of one sample t-test for all pairwise difference the best and the worst. Since even small differences in objective values are crucial for optimization performance, setting d correctly is clearly important – however for individual problems, the performance curve seems to be unimodal, with the best settings not varying much. The setting we used in the main experiments ($\sqrt{|V|}$) appears to have been a reasonable

**Fig. 1.** VLR List Size Comparison: Distribution of Deviation Current Best from Best Known Value every 10^7 Evaluations (Median and Quartiles over 50 Runs) on G37

choice, though a value between $\sqrt{|V|}/2$ and $\sqrt{|V|}$ may have been better. The unimodality of the search means that adaptive measures could be used to choose d , but we leave this for future work.

Figure 1 illustrates the effect of VLR list size. The best solutions found each 10 million evaluations from 50 runs on the G37 form the raw data. The vertical axis indicates the deviation from the best known solution. The dotted lines show the median, and the translucent areas indicate the interquartile ranges (rank statistics are more useful here because the distributions are highly skewed). Larger list sizes consistently lead to better performance (though there is substantial overlap). Larger VLR sizes come with a memory cost, but a VLR size of 8,000 is trivial today; because of the use of the TRIE structure, there is little additional time cost. Thus it seems sensible to use reasonably large VLR sizes, even though the performance gains are relatively small. We conclude that the larger list has better performance since the t-test results for all pairwise difference at the end of the runs are $t = 11.9942$, $p\text{-value} < 2.2e - 16$, and $mean = 0.064$, respectively.

6 Conclusion

The VLH mechanism, inspired by MicroRNA, and extended by the region structure in the VLR algorithm, has shown good performance on the Max-cut problem. VLR's design uses adaptive feedback to maintain the balance between exploration and exploitation through more effective use of information gathered from the search process. We think the performance of VLR derives from its targeted exploration, adapting to the fitness landscape rather than exploring randomly. It thus adopts good features from two important heuristics, namely simulated annealing and Tabu search, combining them in ways that yield useful increments in performance.

In the future we plan to test VLR's performance on a wider range of optimization problems. We also plan to explore the parameter space and algorithm details more deeply – finding ways to determine the optimum list size, and testing whether there are more effective ways to change the EC parameter than the current linear increase and decrease. There is also potential to explore more informed and efficient search operators. Preliminary work in these directions has yielded improved results, but detailed results are not yet available.

Acknowledgements. This work was supported by the Engineering Research Center of Excellence Program of Korea Ministry of Science, ICT & Future Planning(MSIP) / National Research Foundation of Korea (NRF) (Grant NRF-2008-0062609). The ICT at Seoul National University provided research facilities for this study.

References

1. Chen, K., Rajewsky, N.: The evolution of gene regulation by transcription factors and microRNAs. *Nature Reviews Genetics* 8(2), 93–103 (2007)
2. Glover, F.: Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* 13(5), 533–549 (1986)

3. Chang, K.C., Du, D.: Efficient algorithms for layer assignment problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 6(1), 67–78 (1987)
4. Pinter, R.Y.: Optimal layer assignment for interconnect. *Adv. VLSI Comput. Syst.* 1(2), 123–137 (1984)
5. Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research* 36(3), 493–513 (1988)
6. Karp, R.M.: Reducibility among combinatorial problems. In: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) *50 Years of Integer Programming 1958-2008*, pp. 219–241. Springer, Heidelberg (2010)
7. Burer, S., Monteiro, R.D.C., Zhang, Y.: Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization* 12, 503–521 (2000)
8. Festa, P., Pardalos, P., Resende, M., Ribeiro, C.: Randomized heuristics for the max-cut problem. *Optimization Methods and Software* 17(6), 1033–1058 (2002)
9. Martí, R., Duarte, A., Laguna, M.: Advanced scatter search for the max-cut problem. *INFORMS J. on Computing* 21(1), 26–38 (2009)
10. Kochenberger, G.A., Hao, J.K., Lü, Z., Wang, H., Glover, F.: Solving large scale max cut problems via tabu search. *Journal of Heuristics* 19(4), 565–571 (2013)
11. Glover, F., Lü, Z., Hao, J.K.: Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR, Q. J. Oper. Res.* 8(3), 239–253 (2010)
12. Song, B., Li, V.: A hybridization between memetic algorithm and semidefinite relaxation for the max-cut problem. In: *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference, GECCO 2012*, pp. 425–432. ACM, New York (2012)
13. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* 42(6), 1115–1145 (1995)
14. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization* 10, 673–696 (1997)

A Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem with Total Flow Time Criterion

Valentino Santucci, Marco Baidoetti, and Alfredo Milani

Department of Mathematics and Computer Science
University of Perugia, Italy
{valentino.santucci,baidoetti,milani}@dmi.unipg.it

Abstract. In this paper a new discrete Differential Evolution algorithm for the Permutation Flowshop Scheduling Problem with the total flow-time criterion is proposed. The core of the algorithm is the distance-based differential mutation operator defined by means of a new randomized bubble sort algorithm. This mutation scheme allows the Differential Evolution to directly navigate the permutations search space. Experiments were held on a well known benchmark suite and the results show that our proposal outperforms state-of-the-art algorithms on the majority of the problems.

Keywords: Differential Evolution, Permutation Flowshop Scheduling Problem, Randomized Bubble Sort.

1 Introduction and Related Works

The Permutation Flowshop Scheduling Problem (PFSP) is a type of scheduling problem widely encountered in areas such as manufacturing and large scale products fabrication [1]. The goal of PFSP is to determine the best permutation $\pi = \langle \pi[1], \dots, \pi[n] \rangle$ of n jobs that have to be processed through a sequence of m machines.

Here we focus on the Total Flow Time (TFT) criterion that consists in minimizing the objective function

$$f(\pi) = \sum_{j=1}^n c(m, \pi[j]) \quad (1)$$

where $c(i, \pi[j])$ is the completion time of job $\pi[j]$ on machine i and is recursively calculated in terms of the processing times $p_{i, \pi[j]}$ as:

$$c(i, \pi[j]) = \begin{cases} p_{i, \pi[j]} & \text{if } i = j = 1 \\ p_{i, \pi[j]} + c(i, \pi[j-1]) & \text{if } i = 1 \text{ and } j > 1 \\ p_{i, \pi[j]} + c(i-1, \pi[j]) & \text{if } i > 1 \text{ and } j = 1 \\ p_{i, \pi[j]} + \max\{c(i, \pi[j-1]), c(i-1, \pi[j])\} & \text{if } i > 1 \text{ and } j > 1 \end{cases} \quad (2)$$

The PFSP with the TFT criterion has been demonstrated to be NP hard for two or more machines. Therefore, even due to its practical interest, many researches have been devoted to finding high quality and near optimal solutions by means of heuristic or meta-heuristic approaches [1,2]. A report of the state-of-the-art methods for PFSP-TFT has been recently provided in [2] where it is shown that the most performing meta-heuristics are: VNS₄ [3], AGA [4] and GM-EDA together with its hybrid variant HGM-EDA [2]. VNS₄ applies a variable neighborhood search (VNS) to an initial permutation built by means of a constructive heuristic called LR(n/m) [5]. AGA is an asynchronous genetic algorithm hybridized with VNS. GM-EDA is an estimation of distribution algorithm that adopts a probabilistic model for the permutations space known as generalized Mallows model, while HGM-EDA represents the hybridization of GM-EDA with a VNS scheme.

Differential Evolution [6] is one of the many approaches to evolutionary computation [7,8,9]. Although its effectiveness in numerical spaces, DE applications to combinatorial problems, and in particular to permutation-based problems, are still unsatisfactory. To the best of our knowledge, all the DE algorithms for the PFSP proposed in literature (see for example the schemes reported in [10] or the more recent ones [11,12]) adopt some transformation scheme to encode permutations into numerical vectors. This distinction between the phenotypic and genotypic space introduces large plateaus in the numerical landscape and is probably the reason of their poor performances. To address this issue, in this paper we propose a discrete DE scheme for the PFSP-TFT problem that works directly on the permutations space. Since the differential mutation operator has been generally considered the key component of DE [13], our approach mainly relies on a differential mutation operator that directly handles permutations, thus trying to fruitfully bring the DE search properties from the numerical space to the combinatorial space of permutations. Furthermore, a new $O(n^2)$ randomized bubble sort algorithm is provided.

The rest of the paper is organized as follows. The permutation-based differential mutation operator and the new randomized bubble sort algorithm are introduced and motivated in Section 2. The full DE scheme for PFSP-TFT is described in Section 3. An experimental analysis of the proposed approach is provided in Section 4. Finally, conclusions are drawn in Section 5 and some future lines of research are depicted.

2 Differential Mutation in the Permutations Space

Differential Evolution (DE) [6] is a popular and powerful evolutionary algorithm over continuous search spaces using the differential mutation operator as its key component [13]. In the most common variant, for each population individual $x_i \in \mathbb{R}^n$, three different parents $x_{r_0}, x_{r_1}, x_{r_2}$ are randomly selected from the current population and a mutant $v_i \in \mathbb{R}^n$ is generated according to

$$v_i = x_{r_0} + F \cdot (x_{r_1} - x_{r_2}) \quad (3)$$

where the scalar parameter F usually lies in $(0, 1]$. It has been argued that the differential mutation confers to DE the “contour matching” property (term coined by Price et al. in [13]), i.e., it allows DE to automatically adapt both mutation step size and orientation to the objective function landscape.

Here we propose a differential mutation scheme that directly works on the permutations space and that inherits, in some geometric sense, the “contour matching” property of its numerical counterpart.

The permutations of the set $\{1, 2, \dots, n\}$, together with the usual permutations composition operator \circ , form a group denoted by $S(n)$ where each $\pi \in S(n)$ has its inverse, denoted by π^{-1} .

It is possible to bring the classical concepts of sum and difference of \mathbb{R}^n in $S(n)$. Indeed, by defining the sum of $\pi_1, \pi_2 \in S(n)$ as $\pi_1 \circ \pi_2$, their difference can be straightforwardly defined as $\pi_2^{-1} \circ \pi_1$ since $\pi_1 = \pi_2 \circ (\pi_2^{-1} \circ \pi_1)$. Therefore, by temporarily omitting the scale factor F , equation (3) can be rewritten for permutations as:

$$\nu_i = \pi_{r_0} \circ (\pi_{r_2}^{-1} \circ \pi_{r_1}) \tag{4}$$

In order to introduce the scale factor F in equation (4) we need to define an operation which, given a scalar $F \in [0, 1]$, scales down a permutation π to a “truncated” permutation $F \cdot \pi$. A possible approach is to choose a set of generators $G \subseteq S(n)$ and decompose π in the compositions chain $g_1 \circ \dots \circ g_L$ where $g_1, \dots, g_L \in G$. Therefore, by defining $F \cdot \pi = g_1 \circ \dots \circ g_k$, with $k = \lceil F \cdot L \rceil$, it is finally possible to provide a differential mutation for permutations as:

$$\nu_i = \pi_{r_0} \circ (F \cdot (\pi_{r_2}^{-1} \circ \pi_{r_1})) \tag{5}$$

Interestingly, the introduction of a generators set G allows a useful geometric interpretation of the search space. Indeed, given $G \subseteq S(n)$, it is possible to represent the permutations search space as a Cayley graph Γ , i.e., a regular graph whose vertices are the permutations of $S(n)$ and, for any $\pi \in S(n)$ and $g \in G$, the vertices corresponding to π and $\pi \circ g$ are joined by an edge labeled with g . This allows in turn: (1) to derive a metric distance function corresponding to the length of a shortest path between two permutations in Γ , (2) to view the difference between π_1 and π_2 as the compositions chain of the edges labels in a shortest path from π_2 to π_1 in Γ , and (3) to interpret the scaled difference as a truncated shortest path.

However, different sets of generators are possible for $S(n)$. Each one may lead to a different search space structure thus have a different impact on the search algorithm. Here, we consider the three main generators sets of $S(n)$ [15]:

- the set of all transpositions $T = \{(i, j)_T : 1 \leq i < j \leq n\}$, where $(i, j)_T$ denotes the permutation which only swaps the elements at places i and j ,
- the set of all insertions $I = \{(i, j)_I : i \neq j \text{ and } 1 \leq i, j \leq n\}$, where $(i, j)_I$ denotes the permutation that shifts the element at place j to place i ,
- the set of all simple transpositions $ST = \{(i, i+1)_T : 1 \leq i \leq n - 1\}$, i.e., the permutations which only swap two adjacent elements (note that $(i, i+1)_T = (i, i+1)_I = (i+1, i)_I$).

T and I have respectively $\binom{n}{2}$ and $(n-1)^2$ elements, and both produce a search space diameter of $n-1$. Their induced distance functions are known in literature as, respectively, Cayley distance and Ulam distance [15]. Instead, ST , which is a proper subset of both T and I , has $n-1$ elements and provides a diameter of $\binom{n}{2}$. Its induced distance function is known as Kendall- τ distance [15] and equals the number of inversions of either $\pi_2^{-1} \circ \pi_1$ or $\pi_1^{-1} \circ \pi_2$.

The choice among these sets of generators has been made by exploiting the hypothesis that a smoother objective function landscape is a benefit for an evolutionary algorithm. In order to detect which among T , I and ST produces the smoothest landscape on the PFSP-TFT problem, we made two experimental investigations. For several instances of the Taillard benchmark problems (see Section 4) we have generated 10 000 random permutations. For each one, and for $d = 1, \dots, 10$, we have tabulated its TFT relative difference after the application of a random transposition of the type $(i, i+d)_T$ (first experiment) and a random insertion of the type $(i, i+d)_I$ (second experiment). From the box-plots reported in Figures 1a and 1b for the first instance of the Taillard problems 100×5 (other instances have the same behavior) it is possible to deduce that $d = 1$ provides the smoother TFT variation and that this variation increases with d . Therefore, by recalling the fact that both transpositions and insertions reduce to simple transpositions when $d = 1$, the search space for the differential mutation operator has been structured using ST as set of generators.

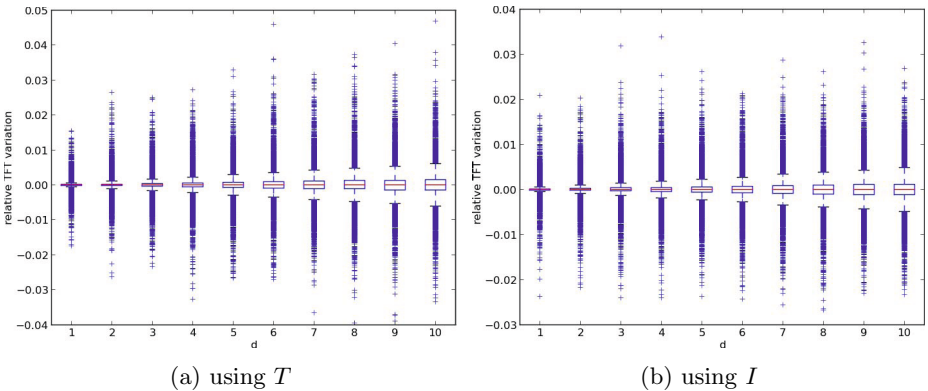


Fig. 1. Relative TFT variations on the first instance of the 100×5 Taillard problems

The truncated permutation $F \cdot \pi$ can be computed using the well known bubble sort algorithm. However, $F \cdot \pi$ is not unique in general because π can have several different shortest representations as compositions chain of simple transpositions. Hence, in order to design a mutation scheme as fair as possible, we propose a randomized version of bubble sort that is outlined in Algorithm 1.

The *RandBS* algorithm sorts the permutation π (and any array of sortable elements) with the optimal number of adjacent swaps. Indeed, at each iteration

Algorithm 1. Randomized Bubble Sort

```

1: function RANDBS( $\pi, n$ ) ▷  $\pi$  is the permutation of degree  $n$  to sort
2:    $CC \leftarrow \langle \rangle$  ▷  $CC$  will be the sequence of simple transpositions that sorts  $\pi$ 
3:    $LST \leftarrow \{i : \pi[i] > \pi[i + 1]\}$ 
4:   while  $LST \neq \emptyset$  do
5:      $i \leftarrow \text{RemoveRandomElement}(LST)$ 
6:     Swap  $\pi[i]$  and  $\pi[i + 1]$ 
7:     Append  $i$  to  $CC$ 
8:     if  $i > 0$  and  $i - 1 \notin LST$  and  $\pi[i - 1] > \pi[i]$  then
9:       Add  $i - 1$  to  $LST$ 
10:    if  $i < n - 1$  and  $i + 1 \notin LST$  and  $\pi[i + 1] > \pi[i + 2]$  then
11:      Add  $i + 1$  to  $LST$ 
12:    end while
13:    return  $CC$ 
14: end function

```

of the while loop: (1) a simple transposition is applied to π , thus reducing by one the Kendall- τ distance to e (the ordered permutation), (2) LST contains exactly the simple transpositions that “move” π towards e . This allows also to limit the number of iterations to $\binom{n}{2} = O(n^2)$. Then, it is easy to prove that the time complexity of *RandBS* is $O(n^2)$ as the one of its classical counterpart.

Furthermore, it is worthwhile to notice that *RandBS* produces, as a second result, CC , i.e., a minimal-length sequence of simple transpositions that sorts π . By reversing the sequence CC , the compositions chain of simple transpositions of π is obtained. Interestingly, CC equals to a sequence of edges labels obtained by a “never go back” random walk from π towards e in the subgraph of Γ composed by the permutations σ such that $d_K(\pi, \sigma) + d_K(\sigma, e) = d_K(\pi, e)$, where $d_K(\cdot, \cdot)$ is the Kendall- τ distance.

Hence, the application of *RandBS* to $\pi_{r_2}^{-1} \circ \pi_{r_1}$ allows to randomly produce one of its decompositions. Then, by truncating it as aforementioned we obtain $F \cdot (\pi_{r_2}^{-1} \circ \pi_{r_1})$ and thus we have a procedure to compute the differential mutation of equation (5).

3 Differential Evolution for Permutations

The Differential Evolution for the Permutations space (DEP), outlined in Algorithm 2, directly evolves a population of NP permutations π_1, \dots, π_{NP} . Its main scheme resembles that of the classical DE with the introduction of a restart mechanism and a memetic local search procedure. Moreover, important variations have been made to the population initialization and to the genetic operators of mutation, crossover and selection. All these components are described in the following.

The population is initialized with $NP - 1$ random permutations and the remaining one is obtained by means of the constructive heuristic $LR(n/m)$ [5].

For each population individual π_i , a mutant permutation ν_i is generated according to equation (5) and using the procedure described in Section 2. In order to avoid the setting of the scale factor F , the self-adaptive scheme proposed in jDE [16] has been used for its online adaptation.

Algorithm 2. Differential Evolution for Permutations

```

1: Initialize Population
2: while evaluations budget is not exhausted do
3:   for  $i \leftarrow 1$  to  $NP$  do
4:      $\nu_i \leftarrow$  DifferentialMutation( $i$ )
5:      $v_i^{(1)}, v_i^{(2)} \leftarrow$  Crossover( $\pi_i, \nu_i$ )
6:     Evaluate  $f(v_i^{(1)})$  and  $f(v_i^{(2)})$ 
7:   for  $i \leftarrow 1$  to  $NP$  do
8:      $\pi_i \leftarrow$  Selection( $\pi_i, v_i^{(1)}, v_i^{(2)}$ )
9:   if restart criterion then
10:    Perform a Baldwinian Local Search on  $\pi_{best}$ 
11:    Restart Population
12: end while

```

The crossover between the population individual π_i and the mutant ν_i is performed according to the two-point crossover version II (TPII) proposed in [7] and used by AGA [4]. Differently from the classical DE crossover, TPII produces two offspring individuals, i.e., $v_i^{(1)}$ and $v_i^{(2)}$. Two indices j, k , such that $1 < j < k < n$, are randomly generated. $v_i^{(1)}[h] = \pi_i[h]$ for $j \leq h \leq k$ and the missing jobs are placed in $v_i^{(1)}$ using the order of their appearance in ν_i . Finally, $v_i^{(2)}$ is filled in the same way but by reversing the role of π_i and ν_i .

In order to choose the trial v_i that will compete with π_i , a preliminary selection between the two offspring individuals is performed according to $v_i = \operatorname{argmin} \{f(v_i^{(1)}), f(v_i^{(2)})\}$.

The new population individual π'_i is chosen by a “biased” selection between v_i and π_i performed according to:

$$\pi'_i = \begin{cases} v_i & \text{if } f(v_i) < f(\pi_i) \text{ or } r < \max\{0, 0.01 - \Delta_i\} \\ \pi_i & \text{otherwise} \end{cases} \quad (6)$$

where r is a random number in $[0, 1]$ and Δ_i is the relative fitness variation $(f(v_i) - f(\pi_i))/f(\pi_i)$. Similarly to classical DE selection, v_i enters the next generation population if it is fitter than π_i . Otherwise, v_i may be selected with a small probability that linearly shades from 0.01 when $\Delta_i = 0$ to 0 when $\Delta_i = 0.01$. This criterion allows: (1) to slow down the population convergence, (2) to reduce the number of restarts, and (3) to mitigate the super-individual effect observed in some preliminary experiments.

Finally, a restart mechanism has been introduced in order to completely avoid the stagnation of the population. When the population fitnesses are the same, the best individual is kept and the other $NP - 1$ permutations are randomly reinitialized. Furthermore, a local search procedure is applied to the best individual using a Baldwinian approach, that is, the result of the local search is collected but does not enter the DEP population. The local search scheme employed is similar to VNS₄ [3] without shakes. A greedy local search using the interchange

neighborhood is carried out until a local minimum is found. Then, the best neighbor in its insertion neighborhood is chosen and the process is iterated until a local minimum for both neighborhoods is reached. Moreover, it is worth to notice that the interchange local search iterates by randomly scanning the permutation components at every step and selecting the first improvement found.

4 Experiments

The performances of DEP have been evaluated on the well known 120 benchmark problems proposed by Taillard in [17]. For each problem instance 20 runs were made and the results have been compared with those provided in [2] for the four PFSP-TFT state-of-the-art methods: AGA, VNS₄, GM-EDA and HGM-EDA. DEP population size has been set to $NP = 100$ after some preliminary experiments and, in order to provide a fair comparison, the same caps of objective function evaluations reported in [2, Table III] have been adopted.

The performance measure employed is the average relative percentage deviation (ARPD):

$$ARPD = \left(\sum_{i=1}^{20} \frac{(Alg_i - Best) \times 100}{Best} \right) / 20 \quad (7)$$

where Alg_i is the final TFT value found by the algorithm in its i^{th} run, and $Best$ is the best known TFT value for the problem instance at hand.

In order to detect the statistical differences between the performances of DEP and each of the other algorithms, as suggested in [18], we applied to every $n \times m$ problem configuration the non-parametric $1 \times N$ Friedman's test and the Finner post-hoc procedure to the average TFT results produced by each algorithm on every instance.

The best TFT values and the ARPDs of each algorithm are reported in Table 1. The TFTs in bold indicates when DEP reaches the best value and the asterisk denotes when it is a new known optimal TFT. Minimal ARPDs are reported in bold.

Furthermore, for each problem configuration the Friedman's average ranking of all the algorithms are provided. Values in bold denote that DEP significantly outperforms the algorithm, while values in italic denote that DEP is significantly outperformed by the algorithm.

In 79 instances over 120, DEP reaches the best TFT, and, most remarkably, in 45 cases they are the new known best values. Moreover, it is worth to notice that DEP has obtained new optima for 23 over 30 instances of size $100 \times m$ and for 18 over 20 instances of size $200 \times m$, which are reputed to be difficult.

The robustness of DEP is proved by the fact that it presents the lowest ARPD results in 96 instances. Again, in almost all 100 and 200 jobs problems, DEP is the best algorithm in average.

Except the case of 500 jobs, DEP has always the lowest Friedman's average rank. The results can be summarized as follows:

Table 1. Experimental Results

Instance	Best	AGA	VNS ₄	GM-EDA	HGM-EDA	DEP	Instance	Best	AGA	VNS ₄	GM-EDA	HGM-EDA	DEP
20 × 5	14033	0.00	0.00	0.18	0.00	0.00	100 × 5	*253605	0.29	1.25	0.87	0.23	0.05
	15151	0.00	0.00	0.48	0.00	0.00	*242579	0.30	1.80	1.08	0.35	0.05	
	13301	0.00	0.00	0.50	0.00	0.00	*238075	0.22	1.49	0.85	0.26	0.07	
	15447	0.00	0.00	0.43	0.00	0.00	227889	0.17	1.29	0.78	0.20	0.06	
	13529	0.00	0.00	0.21	0.00	0.00	240589	0.21	1.29	0.80	0.23	0.02	
	13123	0.00	0.00	0.08	0.00	0.00	*232689	0.32	1.52	0.90	0.28	0.06	
	13548	0.00	0.00	0.79	0.00	0.00	240669	0.15	1.34	1.00	0.34	0.25	
	13948	0.00	0.00	0.18	0.00	0.00	*231064	0.29	1.79	1.06	0.35	0.07	
	14295	0.00	0.00	0.18	0.00	0.00	*248039	0.40	1.66	1.05	0.38	0.09	
12943	0.00	0.00	0.46	0.00	0.00	*243258	0.19	1.44	1.00	0.28	0.07		
Avg Rank	2.5	2.5	5	2.5	2.5	Avg Rank	2.2	5	4	2.7	1.1		
20 × 10	20911	0.00	0.00	0.45	0.00	0.00	100 × 10	*299101	0.43	1.63	1.80	0.44	0.16
	22440	0.00	0.00	0.54	0.00	0.00	*274566	0.60	1.58	2.08	0.69	0.28	
	19833	0.00	0.00	0.31	0.00	0.00	*288543	0.37	1.57	1.74	0.38	0.18	
	18710	0.00	0.00	0.75	0.00	0.00	*301552	0.50	1.79	2.08	0.53	0.18	
	18641	0.00	0.00	0.35	0.00	0.00	*284722	0.61	1.64	1.95	0.54	0.22	
	19245	0.00	0.00	0.77	0.00	0.00	*270483	0.42	1.76	1.83	0.45	0.19	
	18363	0.00	0.00	0.47	0.00	0.00	*280257	0.37	1.58	1.65	0.40	0.25	
	20241	0.00	0.00	0.47	0.00	0.00	*291231	0.49	1.77	2.03	0.61	0.27	
	20330	0.00	0.00	0.27	0.00	0.00	302624	0.36	1.46	1.76	0.41	0.20	
21320	0.00	0.00	0.24	0.00	0.00	*291705	0.48	1.84	1.68	0.50	0.06		
Avg Rank	2.5	2.5	5	2.5	2.5	Avg Rank	2.1	4.1	4.9	2.9	1		
20 × 20	33623	0.00	0.00	0.65	0.00	0.00	100 × 20	*366438	0.80	1.70	2.26	0.67	0.37
	31587	0.00	0.00	0.28	0.00	0.00	*373138	0.55	1.43	2.04	0.58	0.25	
	33920	0.00	0.00	0.04	0.00	0.00	371417	0.47	1.31	1.93	0.36	0.21	
	31661	0.00	0.00	0.28	0.00	0.00	*373574	0.60	1.36	1.92	0.45	0.26	
	34557	0.00	0.00	0.26	0.00	0.00	*369903	0.57	1.35	1.92	0.47	0.19	
	32564	0.00	0.00	0.30	0.00	0.00	*372752	0.51	1.46	2.17	0.42	0.30	
	32922	0.00	0.00	0.61	0.00	0.00	*373447	0.70	1.82	2.19	0.63	0.33	
	32412	0.00	0.00	0.52	0.00	0.00	385456	0.46	1.41	1.96	0.43	0.20	
	33600	0.00	0.00	0.56	0.00	0.00	*375352	0.62	1.52	2.01	0.52	0.41	
32262	0.00	0.00	0.41	0.00	0.00	379899	0.48	1.29	2.05	0.49	0.46		
Avg Rank	2.5	2.5	5	2.5	2.5	Avg Rank	2.8	4	5	2.2	1		
50 × 5	64803	0.05	0.78	0.79	0.12	0.05	200 × 10	1047662	0.48	1.25	1.19	0.17	0.21
	68062	0.06	0.88	0.94	0.12	0.08	*1035783	0.94	1.54	1.49	0.32	0.15	
	63162	0.19	1.21	1.34	0.38	0.21	*1045706	0.66	1.62	1.30	0.32	0.15	
	68226	0.17	1.12	1.27	0.22	0.13	*1029580	0.77	1.65	1.38	0.45	0.12	
	69392	0.09	0.87	0.89	0.15	0.09	*1036464	0.68	1.35	1.37	0.19	0.13	
	66841	0.10	0.80	0.82	0.18	0.04	1006650	0.50	1.36	1.39	0.19	0.23	
	66258	0.03	0.74	0.95	0.07	0.02	*1052786	0.95	1.66	1.23	0.24	0.10	
	64359	0.05	0.89	0.97	0.23	0.05	*1044961	0.62	1.51	1.39	0.25	0.11	
	62981	0.09	0.83	0.81	0.14	0.05	*1023315	0.81	1.61	1.29	0.28	0.24	
*68843	0.15	1.13	1.01	0.29	0.10	*1029198	0.97	1.87	1.48	0.39	0.25		
Avg Rank	1.6	4.2	4.8	3	1.4	Avg Rank	3	4.8	4.2	1.8	1.2		
50 × 10	*87204	0.33	1.12	2.11	0.39	0.18	200 × 20	*1225817	0.72	1.44	1.68	0.34	0.16
	82820	0.22	1.09	2.45	0.60	0.30	*1239246	1.07	1.67	1.66	0.54	0.21	
	79987	0.23	1.07	1.84	0.36	0.22	*1263134	1.08	1.65	1.57	0.48	0.26	
	*86545	0.21	0.94	1.87	0.36	0.16	*1233443	1.25	1.84	1.73	0.58	0.24	
	86450	0.14	0.90	2.02	0.38	0.25	*1220117	1.12	1.79	1.93	0.53	0.17	
	86637	0.13	0.77	1.55	0.29	0.11	*1223238	1.17	1.69	1.69	0.46	0.19	
	88866	0.25	0.89	1.97	0.48	0.42	*1237116	1.03	1.65	1.66	0.64	0.15	
	*86820	0.19	0.95	2.04	0.36	0.01	*1238975	1.25	1.72	1.72	0.51	0.19	
	85526	0.29	1.11	2.10	0.42	0.28	*1225186	1.44	1.91	1.80	0.59	0.14	
88077	0.09	0.76	2.00	0.45	0.42	*1244200	1.16	1.62	1.68	0.52	0.11		
Avg Rank	1.6	4	5	3	1.4	Avg Rank	3	4.5	4.5	2	1		
50 × 20	125831	0.10	0.65	1.76	0.39	0.14	500 × 20	6708053	0.11	0.35	8.90	2.02	1.00
	119259	0.04	0.51	1.58	0.22	0.06	6829668	0.25	0.38	8.58	1.94	0.66	
	116459	0.19	0.73	2.24	0.44	0.28	6747387	0.24	0.41	8.46	2.04	1.07	
	120712	0.22	0.61	1.92	0.34	0.34	6787054	0.26	0.45	8.75	1.89	0.84	
	118184	0.40	0.86	2.30	0.52	0.39	6755257	0.39	0.41	8.72	1.92	0.74	
	120703	0.19	0.62	1.78	0.35	0.16	6751496	0.19	0.42	8.58	2.13	0.32	
	122962	0.38	0.71	2.10	0.47	0.36	6708860	0.27	0.45	9.15	2.05	0.93	
	122489	0.16	0.75	2.24	0.55	0.14	6769821	0.31	0.58	8.62	2.09	0.73	
	121872	0.16	0.76	1.79	0.37	0.12	6720474	0.15	0.46	8.69	1.91	0.96	
124064	0.23	0.90	1.95	0.42	0.29	6767645	0.19	0.44	8.51	2.00	0.86		
Avg Rank	1.5	4	5	3	1.5	Avg Rank	1	2.1	5	4	2.9		

- For problems with 20 jobs, all the algorithms perform the same, except GM-EDA which is significantly worse.
- For problems with 50 jobs, DEP has the lowest average rank values and is significantly better than VNS₄, GM-EDA and HGM-EDA.
- For problems with $n = 100$, DEP has the average rank values very close to 1 and has no clear competitor.
- A similar behaviour is found for problems with 200 jobs, but HGM-EDA, although having a worse average rank and obtaining only two best values over 20, is not significantly worse than DEP.
- The only weakness for DEP is found in problems with 500 jobs, where it is outperformed by AGA and VNS₄. Indeed, we observed that the number of restarts was very small or even zero, thus indicating a low convergence rate probably due to the large diameter of the search space.

The conclusion of this analysis is that DEP can be considered among the state-of-the-art PFSP-TFT algorithms and is the best one on the majority of the benchmark problems.

5 Conclusions and Future Works

In this work, a new discrete Differential Evolution algorithm for Permutation spaces (DEP) has been proposed. The main contribution is the differential mutation operator which is defined by means of a randomized bubble sort algorithm and extends the “contour matching” property of classical DE to the permutations space. Moreover, a randomly biased selection operator that allows to improve the population diversity in order to mitigate the super-individual effect has been proposed.

The experimental results on PFSP-TFT show that DEP outperforms the other state-of-the-art algorithms and found 45 new optimal solutions previously unknown.

Promising lines of research for further improvements will focus on the analysis of the contributions of each single DEP component (mutation, crossover, selection, restart, local search) and the tuning of their parameters.

Furthermore, we are planning to investigate the application of the DEP algorithm to other permutation-based problems (like TSP, QAP, LOP, etc.).

Acknowledgments. This work was partially supported by Italian Ministry of Education, University and Research (MIUR) under the PRIN 2010-11 grant no. 2010FP79LR_003 “Logical methods of information management”, by the University of Perugia, DMI Project “Mobile Knowledge Agents in Evolutionary Environments” and by the services provided by the European Grid Infrastructure (EGI), the Italian Grid Infrastructure (IGI) and the National Grid Initiatives for the Virtual Organization (VO) COMP-CHEM.

References

1. Gupta, J., Stafford, J.E.: Flowshop scheduling research after five decades. *European Journal of Operational Research* 169, 699–711 (2006)
2. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A Distance-based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem. *IEEE Transactions on Evolutionary Computation* 99, 1–16 (2013)
3. Costa, W.E., Goldbarg, M.C., Goldbarg, E.G.: New VNS heuristic for total flow-time flowshop scheduling problem. *Expert Systems with Appl.* 39, 8149–8161 (2012)
4. Xu, X., Xu, Z., Gu, X.: An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization. *Expert Systems with Appl.* 38, 7970–7979 (2011)
5. Liu, J., Reeves, C.R.: Constructive and composite heuristic solutions to the $P//\sum C_i$ scheduling problem. *European Journal of Operational Research* 132, 439–452 (2001)
6. Storn, R., Price, K.: Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Jour. of Global Opt.* 11, 341–359 (1997)
7. Murata, T., Ishibuchi, H., Tanaka, H.: Genetic algorithms for flowshop scheduling problems. *Computers & Ind. Eng.* 30(4), 1061–1071 (1996)
8. Milani, A., Santucci, V.: Community of scientist optimization: An autonomy oriented approach to distributed optimization. *AI Communications* 25, 157–172 (2012)
9. Bairoletti, M., Milani, A., Poggioni, V., Rossi, F.: Experimental evaluation of pheromone models in ACOPlan. *Ann. Math. Artif. Intell.* 62(3-4), 187–217 (2011)
10. Onwubolu, G.C., Davendra, D. (eds.): *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*. SCI, vol. 175. Springer, Heidelberg (2009)
11. Cickova, Z., Stevo, S.: Flow Shop Scheduling using Differential Evolution. *Management Information Systems* 5(2), 8–13 (2010)
12. Li, X., Yin, M.: An opposition-based differential evolution algorithm for permutation flowshop scheduling based on diversity measure. *Adv. Eng. Soft.* 55, 10–31 (2013)
13. Price, K.V., Storn, R.M., Lampinen, J.A.: *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin (2005)
14. Moraglio, A., Poli, R.: Geometric crossover for the permutation representation. *Intelligenza Artificiale* 5(1), 49–63 (2011)
15. Schiavinotto, T., Stutzle, T.: A review of metrics on permutations for search landscape analysis. *Computers & Oper. Res.* 34(10), 3143–3153 (2007)
16. Brest, J., Boskovic, B., Mernik, M., Zumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Trans. on Evol. Comp.* 10(6), 646–657 (2006)
17. Taillard, E.: Benchmarks for basic scheduling problems. *European Jour. of Oper. Res.* 64(2), 278–285 (1993)
18. Derrac, J., Garcia, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1, 3–18 (2011)

A Taxonomy of Heterogeneity and Dynamics in Particle Swarm Optimisation

Harry Goldingay and Peter R. Lewis

Aston Lab for Intelligent Collectives Engineering (ALICE),
Aston Institute for Systems Analytics, Aston University, Birmingham, UK
{goldinhj,p.lewis}@aston.ac.uk

Abstract. We propose a taxonomy for heterogeneity and dynamics of swarms in PSO, which separates the consideration of homogeneity and heterogeneity from the presence of adaptive and non-adaptive dynamics, both at the particle and swarm level. It supports research into the separate and combined contributions of each of these characteristics. An analysis of the literature shows that most recent work has focussed on only parts of the taxonomy. Our results agree with prior work that both heterogeneity, where particles exhibit different behaviour from each other at the same point in time, and dynamics, where individual particles change their behaviour over time, are useful. However while heterogeneity does typically improve PSO, this is often dominated by the improvement due to dynamics. Adaptive strategies used to generate heterogeneity may end up sacrificing the dynamics which provide the greatest performance increase. We evaluate exemplar strategies for each area of the taxonomy and conclude with recommendations.

1 Introduction

There has recently been a sharp rise in interest in heterogeneity of swarms for particle swarm optimisation (PSO). Since early results (e.g. [1]) showed the potential benefit of heterogeneity to PSO, it has been shown to offer a high robustness to unknown problems [2]. In an effort to improve the performance and robustness of heterogeneous PSO variants, more recent work (e.g. [3,4]) has focussed on heterogeneity driven by particle-level adaptation, based on run-time information. However, in this drive to add complexity to PSO by incorporating heterogeneity, behavioural dynamics and run-time adaptation, there is a key question which has not yet been fully addressed: are the observed performance improvements due to better heterogeneity itself, run-time adaptation based on state information, or simply the increase in behavioural dynamics? In this paper we tease out these three components of modern adaptive heterogeneous PSO variants, in order to provide some insight into this question.

Our first contribution is a taxonomy of heterogeneity and dynamics in PSO, into which we place existing PSO variants from prior work. Accordingly we show that most recent research has focussed on only part of the design space arising from our taxonomy, in particular neglecting non-adaptive dynamic PSO in both

heterogeneous and homogeneous cases. Our second contribution is to show that these neglected regions of the taxonomy contain PSO variants which outperform similar adaptive heterogeneous variants. Furthermore, the introduction of dynamics often has a greater impact on performance than the introduction of heterogeneity. Therefore, this paper provides insight into existing PSO variants and makes recommendations for future PSO research.

Montes de Oca et al. [2] describe a heterogeneous swarm as one in which at least two particles differ from each other. They found that heterogeneous swarms typically outperform the worst, and in some cases the best homogeneous swarm on a particular problem. They propose that heterogeneity mitigates the risk of choosing the “wrong” variant of PSO for an unknown problem. They identify three types of heterogeneity: i) *static heterogeneity*, in which particles in a heterogeneous swarm never change their configuration (i.e. behaviour), ii) *dynamic heterogeneity*, in which particles’ configurations change either randomly or according to some predetermined sequence over time, and iii) *adaptive heterogeneity*, where particles’ configurations change based on the state of the swarm; we use these three classes as a starting point for our taxonomy. From their analysis, they conclude that future work should focus on adaptive heterogeneity to improve robust performance of PSO across different problems.

Nipomucino and Engelbrecht [5] define dynamic swarms as those in which particles change their behaviours during the search, also drawing the distinction between static, dynamic and adaptive heterogeneous swarms. PSO variants in the above categories have been proposed by Spanevello and Montes de Oca [6], Li and Yang [7], Engelbrecht [3] and Nipomucino and Engelbrecht [8]. While much of the work on dynamic swarms focuses on heterogeneity, dynamics have also proven useful in homogeneous swarms. In one of the most successful early variants of PSO, Shi and Eberhart [9] proposed varying particles’ *inertial weight* over a swarm’s lifetime. Later variants attempt to improve on this with more complex models. Chatterjee and Siarry [10] propose a non-linear update scheme for inertial weights suggesting, however, that an adaptive mechanism for on-line parameter choice would make their algorithm more robust. Such adaptive algorithms have shown good empirical performance [11–13]. Other homogeneous PSO variants use feedback to choose between a discrete set of behaviour types. Riget and Vesterstrøm [14] propose a variant which monitors diversity in order to prevent premature convergence, switching behaviour when particles are closely clustered. Similarly, Evers and Ghalia [15] propose a variant which performs a one-time update of particle positions when diversity drops below some threshold.

2 Forms of Heterogeneity and Dynamics in PSO

Firstly, we focus on whether or not the swarm is homogeneous or heterogeneous. **Homogeneous** swarms are those in which at each point in time, all particles exhibit the same behaviour as each other. **Heterogeneous** swarms are those in which at some point in time, at least two particles exhibit different behaviours from each other. In this description, we focus on particle update behaviour.

However, our taxonomy can also be used to describe other forms of swarm heterogeneity. We further break down homogeneous and heterogeneous PSO variants according to how the distribution of behaviours in the swarm changes over time:

- In **static** swarms, the behaviour of each particle does not change over time.
- In **constrained dynamic** swarms, there is a stationary proportion of each behaviour, under expectation, between time windows of a predefined size. Therefore, in constrained dynamic homogeneous swarms, the entire swarm might progress through a static sequence of behaviours in synchrony.
- In **dynamic** swarms, the proportion of each behaviour changes over time. In dynamic homogeneous swarms the entire swarm might progress through a sequence of behaviours in synchrony, and this sequence varies over time.
- In **adaptive** swarms, the proportion of each behaviour changes over time in response to the state of the algorithm as perceived by the particles.

The majority of PSO variants use **static homogeneous** swarms. Most recent work on heterogeneity in PSO has focussed on the use of adaptive strategies to generate particle behaviour, and therefore use **adaptive heterogeneous** swarms. However, these results have often been used to argue that heterogeneity of a swarm *per se* is beneficial, despite the characteristics of heterogeneity, particle-level and swarm-level dynamic behaviour and adaptivity being conflated. By dividing both homogeneous and heterogeneous swarms into the above groups, we can study heterogeneity separately from dynamics and adaptation. Table 1 shows a classification of existing literature in terms of the taxonomy. This classification includes the earlier categorisation of PSO variants with update rule heterogeneity due to Montes de Oca et al. [2]. It builds on it firstly by considering work in the context of our expanded taxonomy which accounts for dynamics and adaptivity apart from any heterogeneity present, and secondly by including the significant amount of work on heterogeneous PSO since 2009. It is clear that, despite the recent work on heterogeneity in PSO, a great deal of the space defined by the taxonomy remains to be explored.

3 Experimental Analysis of Heterogeneity in PSO

Table 1 shows that despite the recent work on heterogeneity in PSO, there is a large part of our taxonomy yet to be explored. Next, we present an experimental study which establishes that such exploration would be fruitful. We differentiate between update particles on the basis of parameters and by using two qualitatively different behaviours: Standard PSO [16] and Barebones PSO [17]. These use different information (Standard PSO requires velocity while Barebones PSO does not); however, so that a particle can switch freely between behaviours, we require that particles maintain all information required by either behaviour. We say that this is the particle's **cognitive information**: the set $\mathcal{C}_p(t) = \{\mathbf{x}_p(t), \mathbf{v}_p(t), \mathbf{h}_p(t)\}$ where: $\mathbf{x}_p(t)$ is the particle's **position**, $\mathbf{v}_p(t)$ is the particle's **velocity** and $\mathbf{h}_p(t)$ is the particle's **historic best position**. We assume that the aim of a PSO algorithm is to find an input that minimizes the

Table 1. A classification of existing PSO variants, in terms of the proposed taxonomy

<i>Characteristic</i> → <i>Behaviour</i> ↓	Homogeneous	Heterogeneous
Static	Static homogeneity	Static heterogeneity
	<ul style="list-style-type: none"> – Many PSO variants, including standard PSO [16], barebones PSO [17] etc. 	<ul style="list-style-type: none"> – Static heterogeneous PSO (details not available) [6]. – sHPSO: (random assignment) [3]. – Static heterogeneous PSO (various fixed proportions) [2]. – Predator & prey particles [1]. – Neutral & charged particles [18]. – Fitness-distance-ratio and standard particles [19]. – Quantum particles [20]. – Extra central particle [21].
Constrained Dynamic	Constrained dynamic homogeneity	Constrained dynamic heterogeneity
	<ul style="list-style-type: none"> – None. 	<ul style="list-style-type: none"> – Different maximum velocities after restarts (constrained after initialisation phase) [22].
Dynamic	Dynamic homogeneity	Dynamic heterogeneity
	<ul style="list-style-type: none"> – Inertia weight decay: time-based linear [9] and non-linear [10] update. 	<ul style="list-style-type: none"> – None.
Adaptive	Adaptive homogeneity	Adaptive heterogeneity
	<ul style="list-style-type: none"> – Fuzzy adaptive PSO [11] and fuzzy adaptive informed PSO [13]: inertia weight of entire swarm updated based on fuzzy system. – Adaptive PSO: swarm parameters updated based on evolutionary state estimation [12]. – ARPSO: particles simultaneously switch between two behaviours based on diversity [14]. – RegPSO: all particles perform a one-time position update at low diversity [15]. 	<ul style="list-style-type: none"> – Stagnation threshold [6]. – Difference proportional probability [6]. – dHPSO: win-stay-lose-shift [3]. – pHPSO and pHPSO-lin: inspired by ants [5]. – f_k-PSO: probability of behaviour based on prior performance [8]. – ALPSO: particle-level probability matching [7]. – SLPSO: biased probability matching & super-particle [4]. – Cooperator and defector particles [23]. – Various adaptive heterogeneous parameters (see [2]).

result of a **cost function** f , and so a particle's historic best position is simply the lowest cost position it has visited so far. When updating cognitive information, particles may make use of information from their **neighbourhood**: a set of particles whose states they can observe. In this paper, we assume that all particles neighbour each other, allowing particles to make use of the **global historic best position** $\hat{h}_p(t)$: the lowest cost position discovered by any particle.

In **Standard PSO** a particle p updates its velocity in dimension d as follows:

$$v_{p,d}(t+1) = \eta v_{p,d}(t) + \phi_1 \cdot r_{1,d} (h_{p,d}(t) - x_{p,d}(t)) + \phi_2 \cdot r_{2,d} (\hat{h}_d(t) - x_{p,d}(t))$$

where η is the inertial weight coefficient and $r_{1,d}, r_{2,d}$ are independent random numbers drawn from $U[0, 1]$. Particles are attracted to cognitively and socially

determined positions ($\mathbf{h}_p(t)$ and $\hat{\mathbf{h}}(t)$ respectively) and the constants ϕ_1, ϕ_2 determine the relative importance of these positions. If $\mathbf{h}_p(t) = \hat{\mathbf{h}}(t)$, then the social component of equation 3 is omitted (effectively ϕ_2 is set to 0).

In **Barebones PSO** a particle updates its position in dimension d as follows:

$$x_{p,d}(t+1) \sim N\left(\frac{h_{p,d}(t) + \hat{h}_d(t)}{2}, |h_{p,d}(t) - \hat{h}_d(t)|\right)$$

where $N(\mu, \sigma)$ is the Normal distribution with mean μ and standard deviation σ . Barebones PSO does not make use of a velocity component but, for it to be compatible with standard PSO, we set $\mathbf{v}_p(t+1) = \mathbf{x}_p(t+1) - \mathbf{x}_p(t)$.

We say that a particle's **behaviour** at time t , $b_p(t)$ is the update function it is using at that time. In order for a swarm to be dynamic or heterogeneous its particles must be capable of expressing more than one behaviour. A PSO variant is comprised of a set of update functions and a strategy for selecting between these functions. In our study, we make use of two such **behaviour sets** composed of variants of standard PSO and barebones PSO.

The first behaviour set, **cognitive-biased and social-biased (CBSB)** contains two parametrically different versions of standard PSO. The cognitive-biased function (let $\phi = 0.5 + \log 2$ then $\phi_1 = \frac{5\phi}{3}$, $\phi_2 = \frac{\phi}{3}$, $\eta = \frac{1}{2 \log 2}$) makes more use of cognitive information and is suited to exploration, while the social-biased function ($\phi_1 = \frac{\phi}{3}$, $\phi_2 = \frac{5\phi}{3}$, $\eta = \frac{1}{2 \log 2}$) makes more use of social information and is more suited to exploitation. This set allows us to investigate if an algorithm expressing two behaviours which, intuitively, are suited to performing different search tasks, is capable of improving on the best static homogeneous variant. The second behaviour set, **cognitive-biased and barebones (CBBB)** contains two quantitatively different functions: the cognitive-biased function above and barebones PSO. Unlike the CBSB set, the roles of the two functions during search is not so clearly complementary. This allows us to investigate if any results obtained with CBSB only apply when we have clearly complementary behaviours or whether they apply more generally.

3.1 Exemplar Strategies

We now consider the concrete strategies which particles use to select behaviours. The strategies used are simple exemplars, allowing us to realise the full range of swarm-level characteristics described in section 2. With the exception of the adaptive strategies they are equally applicable to homogeneous and heterogeneous swarms to allow a direct comparison. Particles in **static** swarms, by definition, never update their behaviour and so set $b_p(t+1) = b_p(t)$.

Particles in **constrained dynamic** swarms update their behaviour such that the proportion of behaviours is static over some defined time window. For homogeneous swarms, a constrained dynamic selector must be deterministic. We use a time-based selector which cycles through all possible behaviours, with the time spent using each given by $\boldsymbol{\tau} = (\tau_1, \dots, \tau_{N_u})$ where τ_b is the number of time-steps a particle can spend in behaviour b before switching. After this time has expired, particles deterministically change to the next behaviour b with $\tau_b \neq 0$.

For comparability with the constrained dynamic selector, we use a deterministic time-based **dynamic** selector. Similarly to the inertia weight decay used in [9], it is based on the intuition that certain behaviours are advantageous at the start of search while others are advantageous at the end. τ becomes non-static, given by a linear progression from τ^{start} to τ^{end} based on the fraction of the evaluation budget used. Behaviour is then updated as in the constrained dynamic case. We choose τ such that we have 10 behavioural cycles per run ($10 \cdot \sum_{\tau \in \tau} \tau = \text{budget}$ where *budget* is the swarm's budget of function evaluations) which we have empirically found to be reasonable. Unless specified, $\tau_1 = \tau_2$ in the constrained dynamic case and $\tau_1^{\text{start}} = |\tau|$, $\tau_2^{\text{end}} = |\tau|$ in the dynamic case.

For the above selectors, homogeneity and heterogeneity differ only in **initialization**. Homogeneous swarms are initialized uniformly with the desired initial behaviour while heterogeneous swarms are initialized according to a swarm fraction $\rho = (\rho_1, \dots, \rho_{N_u})$ where ρ_b is the fraction of particles initialized with $b_p(1) = b$. This fraction is analogous to the time vector τ with $\rho_b = \frac{\tau_b}{|\tau|}$.

We use an **adaptive** selector based on *win-stay-lose-shift* [3]. Homogeneous (respectively heterogeneous) particles keep track of the time $v_p(t)$ since the global best (respectively, their historic best) position improved. If this time exceeds a threshold θ then the particle will choose another behaviour uniformly at random. Note that an adaptive selector can only be guaranteed to produce a homogeneous swarm if it acts based on global information. In both the homogeneous and the heterogeneous cases, particles are initialized with $b_p(1) = b$.

3.2 Experimental Set Up

We use the homogeneous and heterogeneous versions of each of the strategies defined previously to represent the areas of our taxonomy. Concrete variants of PSO are created by combining these strategies with the two update function sets. We investigate these variants using the well-known set of test functions described by Hansen et al. [24]. As we are conducting a qualitative investigation rather than attempting to establish the best possible variant, we omit the full set of functions in favour of analysing six functions in more detail: *Sphere*, *Ellipsoidal*, *Rosenbrock*, *Rastrigin*, *Weierstrass* and *Schaffer F7*. The problems have been chosen so that we have three unimodal, three multi-modal, three separable and three non-separable functions. The bounds of the search space for all functions are set to $[-5, 5]^D$, where D is the dimensionality. A trial terminates after an evaluation budget of $D * 1000$. All results are based on an average over 50 trials.

3.3 Experimental Results

Here we summarise key results from our experiments, giving more detailed results in our accompanying technical report [25]. Our initial experiment investigates whether heterogeneity or dynamics are sufficient in themselves to improve upon static homogeneous solutions. We evaluate static homogeneous, static heterogeneous, constrained dynamic homogeneous and constrained dynamic heterogeneous variants using the CBSB and CBBB behaviour sets. Comparing these

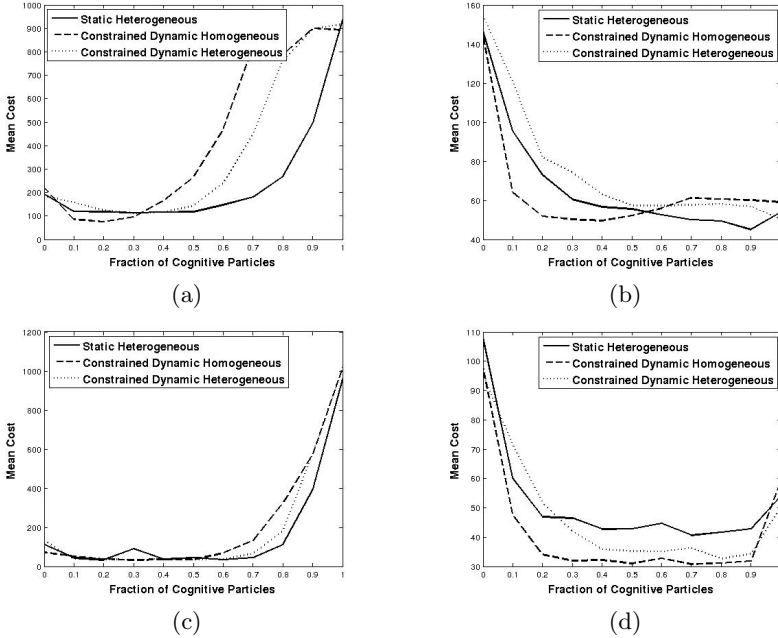


Fig. 1. Mean of best cost across 50 trials as a function of the fraction of cognitive particles in a swarm for the *Rosenbrock* (1a and 1c) and *Rastrigin* (1b and 1d) functions. 1a and 1b use the CBSB update function set, 1c and 1d use the CBBB update function set. Note that with fraction 0 or 1, the swarm exhibits static homogeneity.

strategies allows us to ask whether it is more important that particles express different behaviours over their lifetimes, or are different from each other. We look at the performance of our variants on the 30 dimensional *Rosenbrock* and *Rastrigin* functions, controlling for swarm composition by running the experiment for varying values of ρ and τ (as described in section 3.1) from (0.0, 1.0) (no cognitive-biased particles) to (1.0, 0.0) (all cognitive-biased particles).

The results, shown in figure 1 indicate that heterogeneity and dynamics improve on pure homogeneity, particularly compared to the worst of their two component behaviours. While the type of strategy has a small effect on maximum performance, some strategies are feasible over wider ranges of swarm composition than others. However, a relatively wide range of swarm compositions perform well, confirming results from the literature on heterogeneous swarms and allowing us to draw these same conclusions about constrained dynamic swarms.

To validate the above results more generally and to evaluate the benefits of the dynamic and adaptive models, we test all strategies on 10– 30– and 100– dimensional versions of all evaluation problems. The results in this paper are for the CBBB set, but qualitatively similar results were obtained for the CBSB set, albeit with lower absolute performance. The absolute results do not show a clear pattern by inspection, except for that the best static homogeneous algorithm is typically worse than all heterogeneous or dynamic variants. For simpler analysis

of the variants, we present the median improvement over the best homogeneous case (here defined as $\frac{h-v}{h}$, where h is the mean cost of the best homogeneous variant and v is the mean cost of a variant on a given problem) in table 2.

Table 2. Median improvement of each variant over the *best* static homogeneous case

<i>Characteristic</i> → <i>Behaviour</i> ↓	Homogeneous	Heterogeneous
Static	N/A	21%
Constrained Dynamic	56%	46%
Dynamic	33%	45%
Adaptive	37%	23%

Similarly to our first experiment, all dynamic and heterogeneous strategies improved upon the best static homogeneous behaviour, indicating benefits to dynamics. However, we do not see clear improvements as we move to the more complex areas of our taxonomy: neither our dynamic model of the problem nor our adaptive mechanism (both inspired by successful algorithms from the literature) have improved on constrained dynamic heterogeneous behaviour. In contrast, all non-static strategies are improvements on static ones. Note that we do not claim that a constrained dynamic strategy is optimal (e.g. in comparison to the best possible adaptive strategy), however it strongly indicates that dynamics per-se are making an important contribution to performance and that the benefits of introducing more complex strategies may be outweighed by the loss of dynamics. Even if an adaptive algorithm drives the swarm to some optimal static heterogeneous composition, we have seen from figure 1 that the benefits compared to a sub-optimal but reasonable composition are minimal.

Finally, we investigate the percentage of problems on which adding one level of dynamics/adaptivity improves performance (table 3, note that a figure of 50% indicates equivalent performance). This supports our previous analysis, showing that our dynamic strategy is typically worse than our constrained dynamic, while our adaptive strategy is similar to our dynamic. However, the initial addition of dynamics (static to constrained dynamic) results in a significant improvement. Similarly, heterogeneous variants improve upon their homogeneous counterparts in 60% of tests (fairly uniformly across strategies); it is clear that most of the observed improvements over the static homogeneous case are due to dynamics.

Table 3. The percentage of problems in which moving from one level to the next in our taxonomy of swarm behaviours led to improved performance.

<i>Characteristic</i> → <i>Behaviour comparison</i> ↓	Homogeneous	Heterogeneous
Static to Constrained Dynamic	78%	94%
Constrained Dynamic to Dynamic	17%	28%
Dynamic to Adaptive	61%	50%

4 Conclusions

In this paper we have proposed a taxonomy for heterogeneity and dynamics of swarms in PSO, which acts as a design space. The taxonomy builds upon prior

classifications of heterogeneous PSO variants [2,3], by separating the consideration of homogeneity and heterogeneity from that of adaptive and non-adaptive dynamics, both at the particle and swarm level. It supports research into the separate and combined contributions of these characteristics. In prior work, such questions were difficult to pose, leading to the conflation of the effects of heterogeneity, dynamics and adaptation in some research. An analysis of the literature showed that most recent work focuses on only some regions of the design space; however, other regions may be worthy of more attention. Specifically, while our results agreed with prior work that heterogeneity and dynamics are both useful, with the behaviours we tested, the introduction of dynamics typically had a larger impact on performance than the introduction of heterogeneity. Furthermore, our results show that the recent drive to find optimal forms of heterogeneity at run-time using adaptation may sacrifice the very dynamics which provide the greatest performance increase. It will be important to assess the generality of these conclusions on a wider range of PSO variants and problems.

Our results suggest that future work should focus on dynamics, which have the ability to encode a model of the problem. Furthermore, we believe that there is significant scope for the development of adaptation mechanisms which, rather than adapt particles' behaviours directly, search online for better forms of dynamics which in turn determine behaviour. It also seems appropriate that adaptive PSO variants should not only be compared against static ones, but also against uninformed dynamic variants. Only by doing this can any observed improvement be attributed to the adaptation mechanism and not only dynamics.

References

1. Silva, A., Neves, A., Costa, E.: An empirical comparison of particle swarm and predator prey optimisation. In: O'Neill, M., Sutcliffe, R.F.E., Ryan, C., Eaton, M., Griffith, N.J.L. (eds.) AICS 2002. LNCS (LNAI), vol. 2464, pp. 103–110. Springer, Heidelberg (2002)
2. Montes de Oca, M.A., Peña, J., Stützle, T., Pinciroli, C., Dorigo, M.: Heterogeneous particle swarm optimizers. In: 2009 IEEE Congress on Evolutionary Computation, pp. 698–705. IEEE Press (2009)
3. Engelbrecht, A.P.: Heterogeneous particle swarm optimization. In: Dorigo, M., et al. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 191–202. Springer, Heidelberg (2010)
4. Li, C., Yang, S., Nguyen, T.T.: A self-learning particle swarm optimizer for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 42(3), 627–646 (2012)
5. Nepomuceno, F.V., Engelbrecht, A.P.: A self-adaptive heterogeneous PSO inspired by ants. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Engelbrecht, A.P., Groß, R., Stützle, T. (eds.) ANTS 2012. LNCS, vol. 7461, pp. 188–195. Springer, Heidelberg (2012)
6. Spanevello, P., Montes de Oca, M.A.: Experiments on adaptive heterogeneous PSO algorithms. Technical Report 2009-024, IRIDIA (2009)
7. Li, C., Yang, S.: An adaptive learning particle swarm optimizer for function optimization. In: 2009 IEEE Congress on Evolutionary Computation, pp. 381–388. IEEE Press (2009)

8. Nepomuceno, F., Engelbrecht, A.: A self-adaptive heterogeneous PSO for real-parameter optimization. In: 2013 IEEE Conference on Evolutionary Computation, pp. 361–368. IEEE Press (2013)
9. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: 1998 IEEE Congress on Evolutionary Computation, pp. 69–73. IEEE Press (1998)
10. Chatterjee, A., Siarry, P.: Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & Operations Research* 33(3), 859–871 (2006)
11. Shi, Y., Eberhart, R.: Fuzzy adaptive particle swarm optimization. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 101–106 (2001)
12. Zhan, Z.H., Zhang, J., Li, Y., Chung, H.H.: Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 39(6), 1362–1381 (2009)
13. Neshat, M.: Faipso: Fuzzy adaptive informed particle swarm optimization. *Neural Computing and Applications* 23(1), 95–116 (2013)
14. Riget, J., Vesterstrøm, J.S.: A diversity-guided particle swarm optimizer – the ARPSO. Technical Report 2002-02, Aarhus University
15. Evers, G., Ben Ghalia, M.: Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks. In: IEEE International Conference on Systems, Man and Cybernetics 2009, pp. 3901–3908 (October 2009)
16. Clerc, M.: Standard Particle Swarm Optimisation. Technical Report hal-00764996, HAL (2012)
17. Kennedy, J.: Bare bones particle swarms. In: 2003 IEEE Swarm Intelligence Symposium, pp. 80–87. IEEE Press (2003)
18. Blackwell, T.M., Bentley, P.J.: Dynamic search with charged swarms. In: Genetic and Evolutionary Computation Conference, GECCO 2002, pp. 19–26. Morgan Kaufmann, San Fransisco (2002)
19. Baskar, S., Suganthan, P.N.: A novel concurrent particle swarm optimization. In: 2004 IEEE Congress on Evolutionary Computation, pp. 792–796. IEEE Press (2004)
20. Blackwell, T., Branke, J.: Multi-swarm optimization in dynamic environments. In: Raidl, G.R., et al. (eds.) *EvoWorkshops 2004*. LNCS, vol. 3005, pp. 489–500. Springer, Heidelberg (2004)
21. Liu, Y., Qin, Z., Shi, Z., Lu, J.: Center particle swarm optimization. *Neurocomputing* 70(4-6), 672–679 (2007)
22. Pongchairerks, P., Kachitvichyanukul, V.: Non-homogenous particle swarm optimization with multiple social structures. In: Proceedings of the 2005 International Conference on Simulation and Modeling, pp. 137–144. Asian Institute of Technology, Bangkok (2005)
23. Di Chio, C., Di Chio, P., Giacobini, M.: An evolutionary game-theoretical approach to particle swarm optimisation. In: Giacobini, M., et al. (eds.) *EvoWorkshops 2008*. LNCS, vol. 4974, pp. 575–584. Springer, Heidelberg (2008)
24. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions. Technical Report RR-7215, INRIA (2010)
25. Goldingay, H., Lewis, P.R.: Experimental results concerning heterogeneity and dynamics in particle swarm optimisation. Technical Report AISA-14-01, Aston Institute for Systems Analytics, Aston University, UK (2014)

Derivation of a Micro-Macro Link for Collective Decision-Making Systems

Uncover Network Features Based on Drift Measurements

Heiko Hamann¹, Gabriele Valentini², Yara Khaluf¹, and Marco Dorigo²

¹ Department of Computer Science, University of Paderborn, Paderborn, Germany

{heiko.hamann,yara.khaluf}@uni-paderborn.de

² IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

{gvalenti,mdorigo}@ulb.ac.be

Abstract. Relating microscopic features (individual level) to macroscopic features (swarm level) of self-organizing collective systems is challenging. In this paper, we report the mathematical derivation of a macroscopic model starting from a microscopic one for the example of collective decision-making. The collective system is based on the application of a majority rule over groups of variable size which is modeled by chemical reactions (micro-model). From an approximated master equation we derive the drift term of a stochastic differential equation (macro-model) which is applied to predict the expected swarm behavior. We give a recursive definition of the polynomials defining this drift term. Our results are validated by Gillespie simulations and simulations of the locust alignment.

1 Introduction

Distributed and decentralized systems that rely on self-organization to coordinate a large number of agents are characterized by nonlinear dynamics. These systems rely on positive feedback (amplification), negative feedback (damping), and a multitude of interactions between their components [1]. As a consequence of nonlinearity combined with a large quantity of microscopic details (i.e., features of individual agents), they are generally difficult to analyze and design. Designers may deepen the understanding of these systems by defining appropriate models that reflect specific features of these systems but are “not flooded with microscopic details” [2]. Deriving a macro-model (i.e., a model of swarm features not representing individual agents) mathematically from a micro-model or vice versa is commonly believed infeasible for the general case. In sociology this micro-macro relation is known as the micro-macro link [3,4] that has applications to biology, physics, and engineering, too [5]. An approximation to an actual micro-macro-model for swarm robotics [6] has been proposed [5,7,8] that is capable to represent individual agent trajectories as well as swarm densities. If the considered self-organizing system relies also on inhomogeneous spatial distributions of agents, the modeling task is even more difficult [5,8]. A promising

approach is to use network models as an abstraction of the interaction patterns that emerge from self-organized behaviors (i.e., which agent interacts with which other agents) as done, for example, by Huepe et al. [9] for the example of locusts.

In this paper, we focus on self-organizing collective decision-making (CDM) systems because they generally consist of few, simple control rules and therefore provide a good subject to approach the micro-macro problem under inhomogeneous spatial distributions of agents. CDM systems are found both in natural and artificial swarms. A prominent example in nature is given by CDM in ant colonies [10,11]. CDM systems with tight requirements concerning scalability and robustness are also investigated in swarm robotics [12,13].

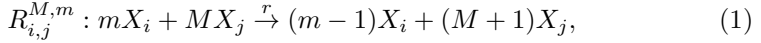
In this paper, we build on a recent work by Biancalani et al. [14]. They define a simplistic collective decision-making model of foraging in ants (meetings of two agents followed by a spontaneous switch of one of them) and investigate noise-induced bistability. Above a critical swarm size, their system fails to converge on a valid collective decision (i.e., it converges on states with conflicting opinions); that is, in contrast to swarm intelligence systems, it does not scale. Nonetheless, their derivation of a macro-model based on the microscopic description of the system behavior, namely chemical reactions, is of particular relevance. We investigate systems operating on local majority rules, that is, subgroups of the swarm cooperate temporarily and have a consensus on the local majority decision. These systems scale [12,13], comply with principles of swarm intelligence, and correspond well to both artificial and natural swarm systems. Following the mathematical approach of Biancalani et al. [14], we derive a stochastic differential equation as a macro-model starting with the microscopic description given by a reaction schema. We focus on the drift terms of these equations that allow to predict the long-term system behavior. Hence, we succeed in establishing a mathematically sound micro-macro link requiring only two minor approximations (Taylor expansion and empirical approach for coefficients in the master equation).

2 Model and Derivation of a Micro-Macro Link

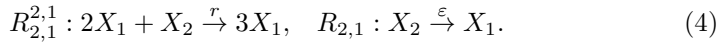
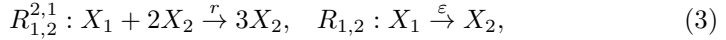
We consider a swarm of N agents undergoing a CDM process. Agents are characterized by their current opinion, for example, their direction of motion or a preference for a particular site in the environment. We restrict our investigations to the simplest case of a binary decision scenario. Each agent favors one out of two possible opinions, henceforth referred to as opinion 1 and opinion 2. Agents change their opinion during the CDM process as they apply a majority rule based on their local neighborhood. That is, when taking a decision, an agent perceives the opinions of its neighbors in a limited perception range, it includes in this group its own opinion, and eventually adopts the opinion favored by the majority of this group. The size G of the neighborhood may vary between different applications of the majority rule. We consider only odd neighborhood sizes to simplify the analysis (i.e., no tie-breakers necessary) and therefore $G \in \mathcal{G} = \{3, 5, \dots\}$. Agents take decisions at a rate r . In addition,

an agent may spontaneously change its opinion at a rate ε . With these spontaneous switches we model noise.

We represent the above described microscopic model using a set of chemical reactions that model all possible causes affecting the opinion of an agent—the reaction schema. Generally, chemical reactions are used to model the dynamics of well-mixed compounds; therefore, our model implicitly assumes a spatially well-mixed system and is thus an approximation of the actual system dynamics. The definition of the reaction schema depends on the particular scenario of interest. We give general equations to define the reaction schema:



Given an agent X_i with opinion $i \in \{1, 2\}$, eq. 1 models the result of the majority rule applied (at a rate r) to a group of $G = m + M$ agents of which a minority mX_i of m agents favor opinion i while a majority MX_j of M agents favor opinion $j \in \{1, 2\} \setminus i$ (thus $m < M$, G odd, transition of one agent from opinion i to j). Eq. 2 describes the spontaneous switch (at a rate ε) of an agent X_i from opinion $i \in \{1, 2\}$ to opinion $j \in \{1, 2\} \setminus i$ and models noise. For clarity, we provide an example of a reaction schema for group size $G = 3$:



Reaction $R_{1,2}^{2,1}$ describes a situation in which an agent with opinion 1 has two neighbors with opinion 2 and, after applying the majority rule, it switches to opinion 2 (respectively, reaction $R_{2,1}^{2,1}$ for an agent with opinion 2). Besides, reaction $R_{1,2}$ models the spontaneous switch in the opinion of an agent with opinion 1 to opinion 2 (respectively, reaction $R_{2,1}$ for an agent with opinion 2).

If we would know the probability density function $f(x_1, x_2, t)$ that describes the time evolution of the proportions of agents x_1 and x_2 (respectively, with opinion 1 and opinion 2), then we would have a complete understanding of the system dynamics. Following the approach of van Kampen [15], $f(x_1, x_2, t)$ is obtained by writing and solving the corresponding master equation

$$\begin{aligned} \partial_t f(x_1, x_2, t) = \sum & [T(x_1, x_2 | x'_1, x'_2) f(x'_1, x'_2, t) \\ & - T(x'_1, x'_2 | x_1, x_2) f(x_1, x_2, t)], \end{aligned} \quad (5)$$

where $x'_1 = x_1 \pm 1/N$, $x'_2 = x_2 \mp 1/N$, and $T(a|b)$ represents the transition rate from state b to state a . However, analytical solutions of master equations are known only for a limited number of cases (cf. van Kampen [15]). Nonetheless, Biancalani et al. [14] derive an approximation to the master equation 5 by means of step operators, which represent the change in the opinion of a single agent, and a Taylor expansion in $1/N$, which yields

$$\begin{aligned} \partial_t f(x_1, x_2, t) \approx & \left[\frac{1}{N}(\partial_{x_2} - \partial_{x_1})T_1 + \frac{1}{N}(\partial_{x_1} - \partial_{x_2})T_2 \right. \\ & \left. + \frac{1}{2N^2}(\partial_{x_1} - \partial_{x_2})^2(T_1 + T_2) \right] f(x_1, x_2, t). \end{aligned} \quad (6)$$

Biancalani et al. [14] reduce eq. 6 to a Fokker-Planck equation by inserting the expressions of the transitions rates T_1 and T_2 followed by rescaling time: $t/N \rightarrow t$. The Fokker-Planck equation is characterized by a drift term, that describes the change in the mean proportions of agents x_1 and x_2 , and a diffusion term, which accounts for the variability of the same quantities. Finally, Biancalani et al. show the equivalence of the obtained Fokker-Planck equation to a system of stochastic differential equations (SDE). We focus on the drift term defined in the system of SDEs because it determines the dominant features of the investigated systems.

The transition rates T_1 and T_2 give the rates at which x_1 and x_2 increase over time. The transition rates depend on the particular reaction schema used to describe the original process. As above, we provide general functions for the corresponding reaction rates of a given reaction schema

$$\begin{aligned} T_1 & \equiv T\left(x_1 + \frac{1}{N}, x_2 - \frac{1}{N} \mid x_1, x_2\right) \\ & \approx \varepsilon x_2 + \sum_{G \in \mathcal{G}} \sum_{n=1}^{[G/2]-1} r \binom{G}{n} x_1^{G-n} x_2^n, \end{aligned} \quad (7)$$

$$\begin{aligned} T_2 & \equiv T\left(x_1 - \frac{1}{N}, x_2 + \frac{1}{N} \mid x_1, x_2\right) \\ & \approx \varepsilon x_1 + \sum_{G \in \mathcal{G}} \sum_{n=1}^{[G/2]-1} r \binom{G}{n} x_1^n x_2^{G-n}. \end{aligned} \quad (8)$$

In eqs. 7 and 8, \mathcal{G} is the set of all possible group sizes while n represents the number of agents in the group favoring the opinion associated to the minority. The binomial coefficients are included based on a heuristic consideration and account for all possible combinations of agents in the group. For the example reaction schema presented above, eqs. 7 and 8 yield the transition rates

$$T_1 = \varepsilon x_2 + r \binom{3}{1} x_1^2 x_2 \quad \text{and} \quad T_2 = \varepsilon x_1 + r \binom{3}{1} x_1 x_2^2. \quad (9)$$

In both reaction rates, the first term models the effect of noise due to spontaneous switching, while the second term models applications of the majority rule.

The approximation of the master equation in eq. 6 together with the transition rates in eqs. 7 and 8 provides a complete macroscopic model derived from the microscopic process described through the reaction schema. As done by Biancalani et al., we reduce the model to a single variable $z = x_1 - x_2$. The change of z over time is given by

$$\dot{z} = \dot{x}_1 - \dot{x}_2 = 2(T_1 - T_2) + D'(x_1, x_2) = 2\Delta_z(T_1, T_2) + D(z). \quad (10)$$

In eq. 10, the drift $2\Delta_z(T_1, T_2)$ and the diffusion $D(z)$ summarize the contributions given by all possible combinations of group sizes and according majorities defined by the reaction schema. Henceforth, we focus on averages $\langle \dot{z} \rangle$ and hence omit the treatment of the diffusion term. The drift term defines the system's main features, such as fixed points and negative/positive feedback. The manual derivation of the term $2\Delta_z(T_1, T_2)$ as a function of z is a rather complex task for nontrivial reaction schemas. Nevertheless, $2\Delta_z(T_1, T_2)$ has a regular structure that consists of a fixed term $-\varepsilon z$, which results from noise, plus a linear combination of polynomials $p_G(z)$ spanning over all considered group sizes G resulting from the majority rule. Linear combinations of such polynomials are easily manageable when applied to the analysis of systems.

The explicit derivation of polynomials $p_G(z)$ for all considered group sizes requires an extensive sequence of change, expansion, and collection of variables. It is a strenuous task whose difficulty increases with the size of the group. We propose a set of recursive functions that automatically generate the corresponding polynomial for a given group size G . The first function

$$p_G(z) = \sum_{m=1}^{\lceil G/2 \rceil - 1} \Delta t_G^m \left(r \binom{G}{m}^2 \right) \quad (11)$$

factorizes the polynomial $p_G(z)$ in a sum of simpler terms Δt_G^m which provide the contribution to the overall drift of a particular group size G and minority m . Function Δt_G^m , that is defined as

$$\Delta t_G^m(\rho) = \rho \left[\frac{1}{4} (1 - z^2) \right]^m h(G - 2m), \quad (12)$$

together with function

$$h(w) = z^w + \sum_{i=1}^{\lceil w/2 \rceil - 1} \Delta t_w^i \left((-1)^{i+1} \binom{w}{i} \right), \quad (13)$$

implement a recursive series of mathematical operations based on the binomial theorem (and related to Pascal's triangle) that are aimed at finalizing the change of variable $z = x_1 - x_2$. The resulting polynomial is characterized by odd powers of z with exponents within $[1, G]$. For the above example, where $G = 3$, the noise term $-\varepsilon z$ plus the recursion of eqs. 11, 12 and 13 yields for the average change

$$\begin{aligned} \langle \dot{z} \rangle &= -\varepsilon z + p_3(z) = -\varepsilon z + \Delta t_3^1 \left(r \binom{3}{1}^2 \right) \\ &= -\varepsilon z + r \binom{3}{1}^2 \left[\frac{1}{4} (1 - z^2) \right] h(1) \\ &= -\varepsilon z + r \binom{3}{1}^2 \left[\frac{1}{4} (1 - z^2) \right] z \\ &= -\varepsilon z + \frac{9}{4} r z - \frac{9}{4} r z^3. \end{aligned} \quad (14)$$

3 Simulation of a Locust Alignment Behavior

The desert locust, *Schistocerca gregaria*, exhibits a collective motion behavior ('marching bands') [16] in which a majority of locusts align and move in a same direction. Individual locusts seem to change their direction of motion as a response to neighbors. In locust experiments [16], the complexity of the natural environment is reduced to a pseudo-1-d setting by using a ring-shaped arena. We use the microscopic model of self-propelled particles proposed by Cziráok et al. [17] as our reference model (henceforth 'Cziráok model').

We study a system of $N = 41$ particles in 1-d space. A particle i has coordinate $y_i \in [0, C)$ (circumference $C = 70$) and discrete, dimensionless velocity $u_i \in [-1, 1]$. We refer to particles with velocity $u_i < 0$ as 'left-goers' (respectively, 'right-goers' for $u_i > 0$). The dynamics of a particle is defined by $y_i(t + 1) = y_i(t) + v_0 u_i(t)$, where $v_0 = 0.1$ is the nominal particle velocity and $u_i(t + 1) = F(\langle u(t) \rangle_i) + \xi_i$ models the particle interaction with its neighbors (subject to noise ξ_i uniformly distributed over $[-\eta/2, \eta/2]$, $\eta = 2.5$). The local average velocity $\langle u(t) \rangle_i$ for the i th particle is calculated over all neighbors located in the interval $[y_i - \Delta, y_i + \Delta]$ for perception range $\Delta = 1.0$. F describes both propulsion and friction forces

$$F(u) = \begin{cases} (u + 1)/2, & \text{for } u > 0 \\ (u - 1)/2, & \text{for } u < 0 \end{cases}. \quad (15)$$

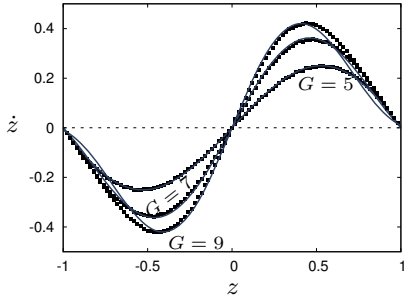
The initial condition is a random uniform distribution for both the particles' coordinates $y_i \in [0, C)$ and their velocities $u_i \in [-1, 1]$.

4 Validation of the Model

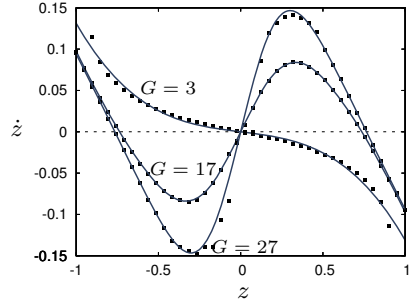
We validate the results presented in Sec. 2 by fitting the drift term defined by linear combinations of polynomials $p_G(z)$ to simulations of two microscopic scenarios. First, for selected group sizes, we fit single polynomials $p_G(z)$ to the average result of simulations of the Gillespie algorithm [18]. Then, we focus on the locust system described in Sec. 3 and we validate our full methodology.

4.1 Gillespie Simulations

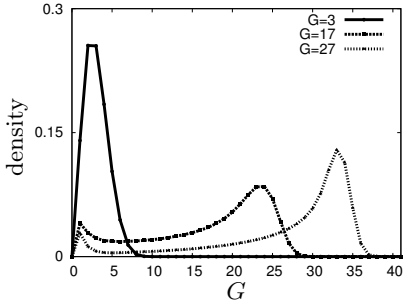
The Gillespie algorithm, also known as Stochastic Simulation Algorithm (SSA), is a Markov chain Monte Carlo method that is proven to generate statistically correct trajectories of a given reaction schema [18]. One of the primary advantages of the Gillespie algorithm is its capability to provide a numerical solution equivalent to that of the master equation by averaging over an ensemble of independent realizations. Given a particular reaction schema, the Gillespie algorithm consists of 3 steps: (i) update the reaction rates for each reaction according to the current state; (ii) randomly determine which and when the next reaction will occur; and (iii) update the system state and jump



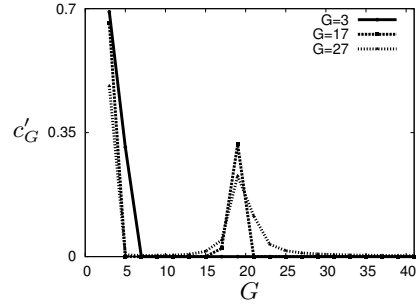
(a) Gillespie, squares: Gillespie simulation (2.5×10^5 samples), line: fitted polynomial



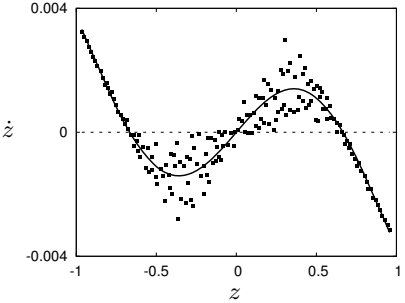
(b) Czirók model, squares: Czirók simulation (10^6 samples), line: fitted polynomials



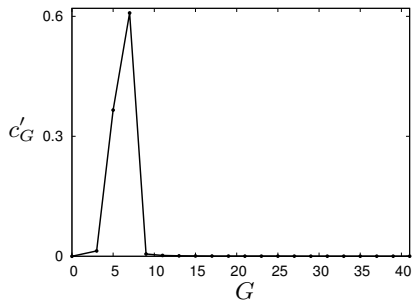
(c) Czirók model, group size distribution measured in simulation



(d) Czirók model, fitted coefficients interpreted as group size distribution



(e) locusts, squares: data from Fig. 3B of [19] (local model of swarm alignment in locusts), line: fitted polynomials



(f) locusts, fitted coefficients interpreted as group size distribution

Fig. 1. Results for fitting the polynomials (eq. 11) to data from Gillespie simulations, simulations of the Czirók model, and to data of swarm alignment in locusts [19]

back to step (i). We run Gillespie simulations for reaction schemas that implement the majority rule for one fixed group size G . In the simulation we measure the probability $P(x_1 + \frac{1}{N} | \text{switch}, x_1)$ that once an agent's switch in opinion is observed it increases x_1 . To get an approximation to the drift term we rescale $\dot{z} \approx P(x_1 + \frac{1}{N} | \text{switch}, x_1) - (2x_1 - 1) = P(z + \frac{1}{N} | \text{switch}, z) - z$. By fitting polynomials $p_G(z)$ to the results of Gillespie simulations we can assess the validity of the approximations introduced in Sec. 2. Fig. 1a shows the results for group sizes $G \in \{5, 7, 9\}$ of the fits between polynomials $p_G(z)$ and the average of 2.5×10^5 Gillespie simulations. We achieve good fits for a range of values $-0.4 < z < 0.4$ but observe systematic deviations for $z < -0.4$ and $z > 0.4$.

4.2 Locust Simulations

In the simulation of the Czirók model we measure the average change $\langle \dot{L} \rangle$ of the ratio of left-goers (averaged over 10^6 independent simulation runs) as a function of the current ratio of left-goers L and the current average neighborhood size G of agents (i.e., agents within perception range Δ) averaged over all agents. These measurements are easily converted to variable z as introduced in Sec. 2 ($z = L - R = 2L - 1$), for the ratio of right-goers R . The measured values of $\langle \dot{z} \rangle$ are then fitted¹ using a sum over the above polynomials $p_G(z)$

$$\langle \dot{z} \rangle = -\varepsilon z + \sum_{G \in \mathcal{G}} c_G p_G(z), \quad (16)$$

with the additional constraints of $c_G \geq 0$ which allow us to interpret the coefficients c_G as weights of each polynomial $p_G(z)$. The results for $G \in \{3, 17, 27\}$ are shown in Fig. 1b. We achieve good fits. In the simulation of the Czirók model we also measure the distributions of neighborhood sizes for given averages of neighborhood sizes (10^6 simulation runs) as shown in Fig. 1c. In Fig. 1d we plot the coefficients, that were obtained in the fitting process for Fig. 1b, in increasing order of G and normalized to $\sum_G c'_G = 1$. By interpreting the coefficients as weights for each neighborhood size G we can read this plot as an approximation of the neighborhood size distribution. Although there is no quantitative agreement, we notice a qualitative agreement. The neighborhood size distribution for $G = 3$ is unimodal as reflected by the coefficients. For $G \in \{17, 27\}$ we have bimodal distributions as in the coefficients. Furthermore, the mean of the fitted coefficients monotonically increases with increasing neighborhood sizes (data not shown).

Finally, we show results for a different source of data. A publication of Yates et al. [19] shows in Figs. 2B and 3B how the drift coefficient depends on the current alignment of a swarm (average velocity). Because the data obtained from experiments with locusts, Fig. 2B in [19], is too noisy, we use instead data from their model, Fig. 3B in [19], to fit our polynomials. The result is a good fit (see Fig. 1e). Fig. 1f shows the corresponding coefficients which peak for

¹ Nonlinear least-squares Marquardt-Levenberg algorithm [20] using gnuplot 4.6 patchlevel 1 (2012-09-26), see <http://www.gnuplot.info/>

neighborhood size $G = 7$ and have a second high value for $G = 5$. Unfortunately, Yates et al. do not report neighborhood sizes. Still, our result seems reasonable given their parameters: locust density $1/3$ and interaction radius 5 . Assuming a uniform distribution, we get neighborhood sizes of about 3.3 . However, we know that locusts align and concurrently tend to cluster. Hence, significantly higher local densities should be expected which supports our finding of neighborhood sizes $G \in [3, 9]$.

5 Discussion and Conclusion

In this paper we extend the approach of Biancalani et al. [14] to reaction equations that include more than two reacting molecules. The obtained method allows to model majority-rule decisions and is applied to CDM systems relevant to swarm intelligence. We report a recursive equation to systematically obtain a set of polynomials. With these polynomials we form linear combinations that define functions of candidate drift terms (i.e., the average change $\langle \dot{z} \rangle$ of swarm fractions that are in favor of one of the opinions). We have shown that, starting from a sample of drift measurements of a particular system, it is possible to obtain a qualitative prediction of the underlying group size distribution by fitting these linear combinations. Our method relies only on measurements of the drift term. Such measurements can be easily obtained as we have shown for the Czirók model and as done for locusts by Yates et al. [19]. Our method applies to both directions of micro-macro transitions: from a given average drift term $\langle \dot{z} \rangle$ (e.g., measured or desired) to an approximation of the underlying group sizes (macro to micro); or vice versa, from a given group size distribution to the prediction of the average drift $\langle \dot{z} \rangle$ (micro to macro). Hence, we establish a micro-macro link.

Motivated by these preliminary results, we plan several extensions. We will investigate methods to apply the master equation instead of approximations while keeping the constraint that the model should be concise and manageable. Alternatively, we will investigate the use of different approximations with the goal of decreasing, for larger group sizes, the difference between the predicted drift term and the results obtained with Gillespie simulations (see Fig. 1a). We will also investigate generalizations of this approach that allow for different decision-making strategies (beyond pure majority decisions) and we will validate the model against a wider set of simulations (e.g., varied perception ranges). We plan to search for polynomials describing CDM systems that are orthogonal functions which can be used as basis functions. This would allow for deterministic calculations of coefficients in the form of a discrete transform (similar to a Fourier transform) instead of the proposed fitting approach.

Acknowledgments. This work was partly supported by the European Research Council through the ERC Advanced Grant “E-SWARM: Engineering Swarm Intelligence Systems” (contract 246939). MD acknowledges support from the Belgian F.R.S.–FNRS.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm Intelligence: From Natural to Artificial Systems*. Oxford Univ. Press, New York (1999)
2. Schweitzer, F.: *Brownian Agents and Active Particles. On the Emergence of Complex Behavior in the Natural and Social Sciences*. Springer, Berlin (2003)
3. Alexander, J.C., Giesen, B., Münch, R., Smelser, N.J. (eds.): *The Micro-Macro Link*. University of California Press, Berkeley (1987)
4. Schillo, M., Fischer, K., Klein, C.T.: The micro-macro link in DAI and sociology. In: Moss, S., Davidsson, P. (eds.) *MABS 2000. LNCS (LNAI)*, vol. 1979, pp. 133–148. Springer, Heidelberg (2001)
5. Hamann, H.: *Space-Time Continuous Models of Swarm Robotics Systems: Supporting Global-to-Local Programming*. Springer, Berlin (2010)
6. Dorigo, M., Birattari, M., Brambilla, M.: Swarm robotics. *Scholarpedia* 9(1), 1463 (2014)
7. Prorok, A., Correll, N., Martinoli, A.: Multi-level spatial models for swarm-robotic systems. *The International Journal of Robotics Research* 30(5), 574–589 (2011)
8. Berman, S., Kumar, V., Nagpal, R.: Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination. In: LaValle, S., et al. (eds.) *IEEE Int. Conf. on Robotics and Automation, ICRA 2011*, pp. 378–385. IEEE Press (2011)
9. Huepe, C., Zschaler, G., Do, A.L., Gross, T.: Adaptive-network models of swarm dynamics. *New Journal of Physics* 13(7), 073022 (2011)
10. Franks, N.R., Mallon, E.B., Bray, H.E., Hamilton, M.J., Mischler, T.C.: Strategies for choosing between alternatives with different attributes: Exemplified by house-hunting ants. *Animal Behavior* 65, 215–223 (2003)
11. Dussutour, A., Beekman, M., Nicolis, S.C., Meyer, B.: Noise improves collective decision-making by ants in dynamic environments. *Proceedings of the Royal Society London B* 276, 4353–4361 (2009)
12. Montes de Oca, M., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., Dorigo, M.: Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making. *Swarm Intelligence* 5, 305–327 (2011)
13. Valentini, G., Hamann, H., Dorigo, M.: Self-organized collective decision making: The weighted voter model. In: Lomuscio, A., et al. (eds.) *Proc. of the 13th Int. Conf. on Autonomous Agents and Multiagent Systems, AAMAS 2014*, pp. 45–52 (2014)
14. Biancalani, T., Dyson, L., McKane, A.J.: Noise-induced bistable states and their mean switching time in foraging colonies. *Phys. Rev. Lett.* 112, 038101 (2014)
15. van Kampen, N.G.: *Stochastic processes in physics and chemistry*. North-Holland, Amsterdam (1981)
16. Buhl, J., Sumpter, D.J.T., Couzin, I.D., Hale, J.J., Despland, E., Miller, E.R., Simpson, S.J.: From disorder to order in marching locusts. *Science* 312(5778), 1402–1406 (2006)
17. Czirók, A., Barabási, A.L., Vicsek, T.: Collective motion of self-propelled particles: Kinetic phase transition in one dimension. *Phys. Rev. Lett.* 82(1), 209–212 (1999)
18. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry* 81(25), 2340–2361 (1977)
19. Yates, C.A., Erban, R., Escudero, C., Couzin, I.D., Buhl, J., Kevrekidis, I.G., Maini, P.K., Sumpter, D.J.T.: Inherent noise can facilitate coherence in collective swarm motion. *Proc. Natl. Acad. Sci. USA* 106(14), 5464–5469 (2009)
20. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics* 11(2), 431–441 (1963)

Messy Coding in the XCS Classifier System for Sequence Labeling

Masaya Nakata¹, Tim Kovacs², and Keiki Takadama¹

¹ The University of Electro-Communications, Japan
m.nakata@cas.hc.uec.ac.jp, keiki@inf.uec.ac.jp

² University of Bristol, UK
kovacs@cs.bris.ac.uk

Abstract. The XCS classifier system for sequence labeling (XCS-SL) is an extension of XCS for sequence labeling, a form of time-series classification where every input has a class label. In XCS-SL a classifier condition consists of some sub-conditions which refer back to previous inputs. Each sub-condition is a memory. A condition has n sub-conditions which represent an interval from the current time t_0 to a previous time t_{-n} . A problem of this representation (called interval coding) is, even if only one input at t_{-n} is needed, the condition must consist of n sub-conditions to refer to it. We introduce a messy coding based condition where each sub-condition messily refers to a single previous time. Unlike the original coding, the set of sub-conditions does not necessarily represent an interval, so it can represent compact conditions. The original XCS-SL evolutionary mechanism cannot be used with messy coding and our main innovation is a novel evolutionary mechanism. Results on a benchmark show that, compared to the original interval coding, messy coding results in a smaller population size and does not require as high a population size limit. However, messy coding requires more training with a high population size limit. On a real world sequence labeling task messy coding evolved a solution that achieved higher accuracy with a smaller population size than the original interval coding.

1 Introduction

Time-series classification has attracted great interests in machine learning. As a kind of time-series classification, *sequence labeling* [3] has been applied in a wide range of real world applications, such as part of speech tagging [8] and recognition of human activity [2]. While typical time-series data is a sequence of values which share a class, sequence labeling data is a sequence of input/class pairs. For example, in a sequence such as speech tagging data, each word in a sentence is one input and is classified as a noun, verb etc. In sequence labeling, the class of the current input may depend on previous inputs, hence a learner may need to refer back to the previous inputs¹. Our interest is in learning human-readable

¹ The class of sequence labeling data may also depend on future inputs. However, in many tasks, such as online learning, we want to predict the current or future class strictly from the past, and so we consider only previous inputs.

(simple, understandable and compact) solutions from sequence labeling data. Learning Classifier Systems (LCSs) evolve general condition-action rules (called *classifiers*). Previously, we introduced XCS for sequence labeling (XCS-SL) [6], an extension of XCS [10]. In XCS-SL a classifier has a *variable-length* condition which consists of sub-conditions C_0, \dots, C_{-n} as memories that refer back to previous inputs. The condition can *grow* and *shrink* by evolution to find a suitable memory size (i.e., the number of sub-conditions which are needed). This variable-length condition is more useful than a *fixed-length* condition (a fixed memory size), since it can handle a larger memory size than the fixed-length one [6]. Some related LCS works, e.g., XCSM [4] which also uses memory and CCS [9] which uses a *chain* of classifiers as a kind of variable length condition, have been presented but due to lack of space we must leave comparison of them and XCS-SL for future work. Similarly we must leave comparison with other sequence labeling algorithms for future work.

The variable-length condition $C = \{C_0, C_{-1}, \dots, C_{-n}\}$ represents an interval of inputs from the current time t_0 to a previous time t_{-n} . However such coding (called *interval coding*) still has a limitation in representing compact conditions. That is, even if only one previous input at t_{-n} is needed to classify the current input at t_0 , the condition must consist of all n memories from C_{-1} to C_{-n} . For example, if we only need the memory at time t_{-n} the minimal condition would be $C = \{C_0, C_{-n}\}$ (or even just $\{C_{-n}\}$ if C_0 is not needed to disambiguate the current input). But, the condition in the interval coding contains all memories.

We introduce a *messy-coding* based condition for XCS-SL. In the new condition, each sub-condition *messily* refers to a single different previous time, hence the condition is not necessarily an interval. For instance, a condition can be $C = \{C_0, C_{-5}, C_{-n}\}$. The messy-coding can remove redundant sub-conditions, so it can represent the minimal conditions. Accordingly, the most important thing when evolving classifiers is probably finding *where* and *how many* previous inputs are needed. To do so we present a novel evolutionary mechanism for messy coding in XCS-SL. We test XCS-SL with messy coding on a benchmark problem (the Layered Multiplexer Problem) and a Activity of Daily Living (ADL) recognition problem [7] as a real world application of sequence labeling.

2 Messy Coding in Sequence Labeling

This section describes sequence labeling in more detail by showing example data. We also explain a difference between the messy coding and the original interval coding (i.e., the variable-length condition) in XCS for sequence labeling.

2.1 Sequence Labeling

As shown in Figure 1, the sequence labeling dataset which is a part of a human-activity recognition can be represented as $\langle \text{time}, \text{input}:\text{class} \rangle$. The input “kitchen” is placed at different time stamps “1pm” and “7pm” but it has different classes “lunch” or “dinner” respectively. Note we do not use the time stamps except to

$\langle 9am, office:work \rangle, \langle 1pm, kitchen:lunch \rangle, \langle 6pm, living:TV \rangle, \langle 7pm, kitchen:dinner \rangle$

Fig. 1. Example dataset of sequence labeling

order the inputs, i.e., a classifier cannot be represented as “IF time is 7pm THEN *dinner*”. Hence, the input “*kitchen*” does not unambiguously identify the current class, i.e., the input is *perceptually aliased*². However, when a learner refers back to the previous input, it can successfully classify it when it considers current and previous inputs. For instance, a minimal condition for correctly predicting the “*dinner*” class in Figure 1 can be $\{(living, t_{-1})\}$. While a minimal condition should consist of minimum elements, many accurate but not minimal conditions can exist such as the condition $\{(kitchen, t_0), (living, t_{-1}), (kitchen, t_{-2})\}$.

A difficulty of sequence labeling is that a learner does not know *where* and *how many* previous inputs are needed to classify the current input. The learner explores many possible conditions to find minimal conditions, hence it may need many memories to refer back to previous inputs at different time stamps.

2.2 Messy Coding vs. Original Interval Coding

The original interval coding (i.e., the variable-length condition) and the messy coding both are a memory-based approach for classifier conditions. In the original interval coding, the condition consists of sub-conditions as a memory, and includes a sub-condition C_0 for the current input at t_0 . This is because, for non-aliasing inputs, classifiers consist of only one sub-condition for the current input. For instance, in Figure 1, classifiers using the original interval coding can be:

$$cl_1 = \{(\#, t_0), (living, t_{-1}): dinner\} \quad cl_2 = \{(\#, t_0), (living, t_{-1}), (\#, t_{-2}): dinner\}$$

Here, *don't care symbol* $\#$ can be any symbol. While the classifier cl_1 has a minimal condition *in the original interval coding*, cl_2 does not since it includes a redundant sub-condition $(\#, t_{-2})$. However, cl_1 is also *not* minimal in Figure 1 because it has $(\#, t_0)$ and we saw in Figure 1 that only $\{(living, t_{-1})\}$ is needed.

In the messy coding, the condition also consists of sub-conditions, but each sub-condition *messily* refers to a different time stamp. Accordingly, unlike the interval coding, the condition using messy coding may have no sub-condition for the current input. For instance, classifiers using the messy coding can be:

$$cl_3 = \{(living, t_{-1}), (kitchen, t_{-2}): dinner\} \quad cl_4 = \{(living, t_{-1}): dinner\}$$

cl_3 does not have the minimal condition in Figure 1 because of $(kitchen, t_{-2})$; cl_4 has the minimal one. So the messy coding can represent more compact conditions than the interval coding. However, the messy coding makes many possible conditions which do not exist in the interval coding. Hence, an evolution of classifiers is important in finding the minimal conditions. We note there are many

² This work does not use a history of previous classes, since our interest is in online-learning where we do not know if the actions were correct in unlabeled data and so the class history may be unsure information.

minimal conditions in the messy coding which may result in many overlapping classifiers. For example, in Figure 1, a condition $\{\textit{kitchen}, t_{-2}\}$ is also the minimal condition for correctly predicting “*dinner*”. These overlapping classifiers should increase the population size but it is unclear whether they will otherwise affect the performance of XCS-SL. We do not consider this issue further.

3 XCS-SL Classifier System

This section describes the mechanism of XCS-SL [6]. XCS-SL almost works the same as standard XCS [1] but some mechanisms in the performance and the discovery components are modified (see [6]). We also explain *subsumption* [1] for the interval coding and the *shrinker*, which can help to find compact conditions.

XCS-SL Classifiers. A classifier in XCS-SL is the same as the standard XCS classifier [1] but it has a new memory size parameter m to determine the number of sub-conditions in its condition $C_0, C_{-1}, \dots, C_{-m}$. Each sub-condition C_{-n} corresponds to the input at the time stamp t_{-n} . The memory size m is determined and fixed when the classifier is generated but the maximum memory size M for all classifiers is set to a fixed value.

Performance Component. The population $[P]$ is initially empty. At the current time t_0 , XCS-SL stacks the current input to the *input list*. When the number of inputs in the list is larger than M , XCS-SL deletes the input at the oldest time stamp t_{-M} in the list. Next, XCS-SL builds a *match set* $[M]$ containing the classifiers in $[P]$ whose sub-conditions C_{-n} each match the stacked input at the corresponding time t_{-n} . If $[M]$ does not contain all the possible actions *covering* [1] generates classifiers; their memory size m is set uniform randomly but the maximum value is the number of inputs in the input list (so it does not have more memory than there are past time steps). Each sub-condition C_{-n} is copied from the corresponding input at the time stamp t_{-n} but each element of the sub-condition is replaced by $\#$ with a probability $P_{\#}$. From here, XCS-SL works the same as XCS in the performance component (see [1]). After that, the reinforcement component [1] is performed the same way as in XCS.

Discovery Component. XCS-SL evolves classifiers using a Genetic Algorithm (GA). In sequence labeling, each input can have its own suitable memory size (i.e., each input may need a different number of previous inputs). Hence, XCS-SL is required to evolve classifiers which have the suitable memory size. Accordingly, XCS-SL builds *subsets* $[A(t_{-n})]$ of the action set which each consists of classifiers in $[A]$ whose memory size m is equal to n . Then XCS-SL selects one *subset* from among the subsets $[A(t_0)], \dots, [A(t_{-M})]$ to perform the GA on. Selection is done by a roulette wheel on the average fitness of each subset. After selection, the GA is applied to classifiers in the selected subset and generates two new offspring with the same memory size as their parents. Evolution finds classifiers with a suitable memory size because classifiers with enough memory have higher fitness than classifiers with too little memory. Classifiers with more memory than they need

also have high fitness, but subsumption removes them. Two offspring are generated as copies of two selected parents and the crossover and mutation operators are applied to the offspring with probabilities χ and μ respectively. In crossover, each sub-condition is recombined with the corresponding sub-condition of the other offspring. The mutation changes elements in each sub-condition, after that it also changes the memory size m of a classifier to a random value with probability μ . If the memory size *shrinks*, the extra sub-conditions C_{-n} ($n > m$) are removed. If the memory size *grows*, new sub-conditions C_{-n} ($n > m$) are added which are copies of the corresponding input at the time stamp t_{-n} in the input list and they are generalized as in covering.

Subsumption and Shrinker. Subsumption is a generalization operator that helps to decrease the population size by subsuming a classifier to a more general classifier. In XCS-SL, subsumption applies to classifiers which have different condition lengths from each other. To compare the generality of these classifiers, we assume the shorter classifier has extra virtual maximally general sub-conditions (that have only #) to fit the condition length of the longer classifier. For instance, as shown below, to compare the generalities of the classifiers cl_a and cl_b , we consider that cl_a has two maximally general sub-conditions “###” added. Accordingly, the sub-conditions C_{-1} and C_{-2} of cl_a are more general than the corresponding sub-conditions of cl_b , hence, cl_a is more general than cl_b .

$$\begin{aligned} cl_a &= \{(1\#0, t_0)\} && \rightarrow \{(1\#0, t_0), (\###, t_{-1}), (\###, t_{-2})\} \\ cl_b &= \{(1\#0, t_0), (10\#, t_{-1}), (11\#, t_{-2})\} && \rightarrow \{(1\#0, t_0), (10\#, t_{-1}), (11\#, t_{-2})\} \end{aligned}$$

Shrinker is a compaction operator that helps to find compact conditions; it decreases the memory size of classifiers whose sub-conditions are maximally general. Specifically, if the sub-condition C_{-m} for the oldest time stamp is coded by only #, then C_{-m} is removed and the memory size m is decreased by 1. This process is repeated recursively. For instance, as shown below, the sub-condition C_{-2} of classifier cl_c is removed, since C_{-2} is the maximally general condition “###”, and the memory size of cl_c is reduced to 1. Note that the shrinker is not applied to classifiers which consist of only sub-condition C_0 . The shrinker is applied to classifiers which are generated by covering and the GA.

$$cl_c = \{(1\#0, t_0), (\#1\#, t_{-1}), (\###, t_{-2})\} \rightarrow \{(1\#0, t_0), (\#1\#, t_{-1})\}$$

4 Messy Coding in the XCS-SL Classifier System

This section presents a modified XCS-SL with messy coding (XCS-SL-messy). Normally in LCS, conditions are fixed-length ternary strings from $\{0, 1, \#\}$. Lanzi [5] introduced messy coding for LCS, in which the # is not represented, and the position of 0s and 1s are explicit. For example, the normal ternary condition $\{1\#0\}$ is equivalent to $\{(1,0), (0,2)\}$ in messy coding. We use a different kind of messy coding. Lanzi encoded conditions on the single current input messily; he did not use memory. In contrast, we encode memories messily: we do not represent fully general memories ($\###$), but we do represent the time-stamp

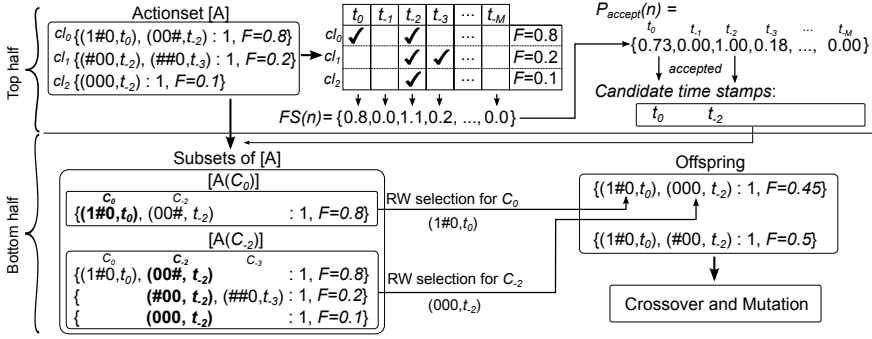


Fig. 2. Discovery component of XCS-SL-messy

of memories which are not fully general e.g., $\{(1\#0, t_0), (11\#, t_{-2})\}$. Note that when a memory is not fully general we use a normal ternary string.

The original XCS-SL evolves a suitable memory size for a classifier but with messy coding we evolve not only how much memory but where (which time steps a classifier refers to). Because we have changed the representation we also have to change the discovery component. This is our most important contribution and we explain it next. We also explain the covering and the shrinker operators which are also modified for the messy coding. Note a classifier for the messy coding is the same as the original XCS-SL except for the condition, which consists of sub-conditions, which each messily refer to a different time stamp.

Covering. When covering takes place, XCS-SL-messy generates classifiers using the messy coding. Firstly, their memory size m is set uniform randomly to determine *how many* sub-conditions are generated. Next, for each sub-condition C_{-n} , a time stamp t_{-n} is set to a random value to determine *where* its sub-condition refers. The time stamp is set except for values which are already assigned in other sub-conditions. The maximum value for the memory size and the time stamp is the number of inputs in the input list.

Discovery Component. We introduce a heuristic to estimate how many and where previous inputs are needed to classify the current input to the correct class. Figure 2 shows an overview of the discovery component we introduced. As shown in the top half of Figure 2, we firstly calculate a *fitness summation* $FS(n)$ for each time stamp t_{-n} . Here, we assume a sub-condition of a classifier which has high fitness is a key memory to disambiguate the current input. $FS(n)$ is calculated by Equation (1), which is a summation of fitness of classifiers which have sub-condition C_{-n} . In Equation (1), $cl_k \in [A](C_{-n})$ denotes classifiers cl_k in $[A]$ which their conditions have sub-condition C_{-n} . Next, an *acceptance probability* $P_{accept}(n)$ is calculated by Equation (2), which is the normalized value of the fitness summation. Next, *candidate time stamps* are selected from among all possible time stamps. For each time stamp t_{-n} , we decide either to accept it as a candidate time stamp with the probability of $P_{accept}(n)$ or to reject it.

$$FS(n) = \sum_{cl_k \in [A](C_{-n})} F_k \quad (1) \quad P_{accept}(n) = \frac{FS(n)}{\max_n FS(n)} \quad (2)$$

After that, XCS-SL-messy generates offspring based on the candidate time stamps. As shown in the bottom half of Figure 2, like the original XCS-SL it builds *subsets* of the action set, but they are built in a different view point from the original one. Specifically, the subset $[A(C_{-n})]$ consists of classifiers in $[A]$ whose conditions include the sub-condition C_{-n} . Next, the offspring are generated from the classifiers in the subsets. The offspring is given a sub-condition for each candidate time stamp. Firstly, for each candidate time stamp t_{-n} , one parent is selected from the corresponding subset $[A(C_{-n})]$. The sub-condition C_{-n} of the offspring is generated as a copy of C_{-n} of the selected parent. This process repeats two times to generate two offspring. The parameters of offspring are set to averages of the corresponding parameters of their parents. The crossover is the same way as the original XCS-SL. The mutation changes the memory size m of a classifier to a random value with probability μ . If the memory size shrinks, the sub-conditions C_{-n} are randomly selected and removed. If the memory size grows, new sub-conditions C_{-n} are added but their time stamp t_{-n} is randomly selected except for time stamps which are already assigned in other sub-conditions. The C_{-n} are copies of the corresponding input at t_{-n} in the input list which are generalized as in covering.

Shrinker. In XCS-SL-messy, if the sub-condition C_{-n} for *any* time stamp t_{-n} is coded only by #, then its sub-condition is removed and the memory size m is decreased. Note in the original XCS-SL, the shrinker takes place *only* on the sub-condition C_{-m} at the oldest time stamp t_{-m} .

5 Experiment on Benchmark Problem

In the well-known family of l -bit Boolean multiplexer functions [10], the first k bits are converted to a decimal index into the remaining bits and the value of the string is the value of the indexed bit. E.g., with $l=6$, the class of 110001 is 1 as the first 2 bits index the final bit. We introduced the n -Layered l -bit Multiplexer Problem (n - l LMP) in [6] as a sequence labeling task. We make a list of D random l -bit binary strings. To train the learner we iterate through them, using one string as input on each time step t_0, t_1, \dots, t_D . In the LMP, the class of the current input may depend on another input. Specifically, the first n bits of the current input are converted to a decimal number as a *reference time* rt . To determine the class of the current input, the LMP refers to the input at t_{-rt} and computes the normal l -bit multiplexer function on it. If the reference time would be negative, i.e., $t_{-rt} < t_0$, we wrap around to the end of the dataset and use t_{D-rt} as the reference input. For instance, on 3-6LMP, for the sequence of inputs $\{\dots, 000000, 001000, \dots\}$, the correct class of the "000000" is 0 since the class is determined by own input due to $rt=0$ (and $\text{index}=0$); the correct class of the "001000" is determined by the previous input "000000" due to $rt=1$ (which means the class is referred to the last input). We use a reward of 1000

for a correct action, otherwise 0. We use the 0-6 and 3-6 LMP with $D=50,000$. Note a 0- l LMP is the normal l -bit multiplexer. Note also the minimal condition *in interval coding* is $\{C_0, \dots, C_{-rt}\}$, but *in the messy coding* it is $\{C_0, C_{-rt}\}$.

5.1 Results

Each experiment consists of a number of problems that the system must solve. In each problem as one iteration, LCS alternatively solves a *learning* problem and an *evaluation* problem (see [10]). We use the standard parameter settings [1]: $\epsilon_0=1$, $\mu=0.04$, $P_{\#}=0.33$, $\chi=0.8$, $\beta=0.2$, $\alpha=0.1$, $\delta=0.1$, $\nu=5$, $\theta_{GA}=25$, $\theta_{del}=20$, $\theta_{sub}=20$, $M=8$, and Action set subsumption and GA subsumption are turned on. We use different population size limits $N=60,000$ and $6,000$. The maximum iteration is 2,000,000. The *performance*, which is the rate of correct actions the LCS executed, and *population size*, which is the number of (macro) classifiers [10], are reported as the moving average of 50,000 evaluation problems. All the plots are averages over 30 experiments.

Figure 3 shows the performances and the population sizes of XCS-SL and XCS-SL-messy on $\{0, 3\}$ -6LMP with $N=60,000$ and $6,000$. From Figure 3 a), with $N=60,000$, on $\{0, 3\}$ -6LMP both systems reach 100% performance, but XCS-SL learns faster than XCS-SL-messy. In contrast, from Figure 3 b), with a small population size limit ($N=6,000$), XCS-SL performs *worse* than XCS-SL-messy: while XCS-SL fails to reach 100% due to the small population size limit, XCS-SL-messy successfully reaches it. While one algorithm outperforms the other depending on the population size limit, for both size limits, XCS-SL has many more classifiers than XCS-SL-messy. Specifically, on 3-6LMP with $N=60,000$, XCS-SL has 9367 classifiers but XCS-SL-messy has 2291.

In summary, results suggest 1) messy coding has a smaller population size (i.e., a number of classifiers in population) than interval coding, 2) interval coding requires a larger population size limit to reach full accuracy, but 3) messy coding is slower to reach full accuracy when the population size limit is large. It is not clear why messy coding results in a smaller population size, but the smaller population explains observation 2) – because interval coding has a larger population size it needs a larger population size limit to function. We hypothesize that 3) is the case because it takes longer to search the larger space of messy classifiers than the smaller space of interval classifiers. Also, the larger population found with interval coding is searching more of the rule space in parallel than XCS-SL-messy’s smaller population.

Results on the layered and regular (i.e., 0- l LMP) multiplexers are similar: the performance of interval coding reaches maximum faster than messy coding, but messy coding has a smaller population size.

6 Experiment on ADL Recognition

This section tests XCS-SL-messy on a real world Activity of Daily Living (ADL) recognition problem, which has the challenge of a small number of instances and

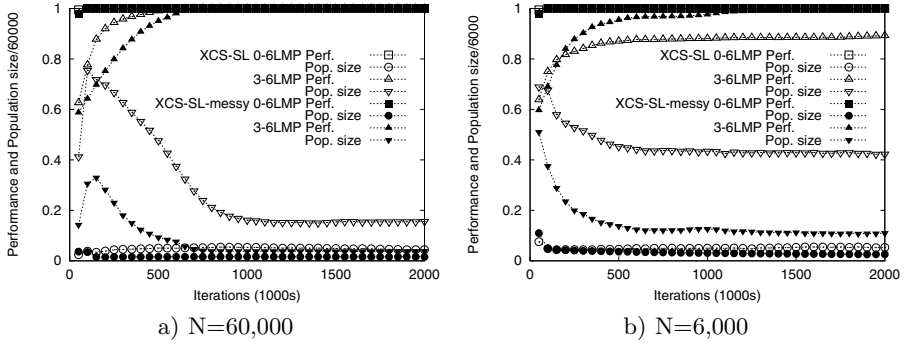


Fig. 3. Performances and Population sizes on $\{0, 3\}$ -6LMP

a large number of classes. ADL recognition [7] is a classification task to recognize human activity from binary sensors. We modify the data (OrdonezA) to be a sequence labeling task. The format of each data point is a time/input/class; an input in the form of binary sensor data consists of three elements (*sensor*, *sensor type* and *room*); a class indicates a human activity. The *sensor*, *sensor type* and *room* can be one of 12 sensors, 5 sensor types and 5 rooms respectively; the class can be one of 10 human activities (see [6], [7]). The dataset has 397 data points.

We use the first 70% as training data and the last 30% as test data. Each experiment consists of a *learning* phase and a *test* phase. The test phase happens after the learning phase. During the test phase, the system must solve the test data, and it does not apply the reinforcement and discovery components. We compare XCS, XCS-SL and XCS-SL-messy, and we employ the same parameter settings of the previous test except for $N = 5000$, the maximum iteration is 200,000 and Action Set subsumption was turned off to avoid overly strong generalization pressure. We calculate the *classification accuracy* and the population size during the test phase, which is the average over 30 experiments.

Table 1. a) Classification accuracies (the top half) and p -values (the bottom half) on ADL recognition. b) Population sizes (the top half) and p -values (the bottom half). Bold text indicates a significant difference ($p < 0.01$).

a) Classification accuracies				b) Population sizes		
	XCS	XCS-SL	XCS-SL-messy	XCS	XCS-SL	XCS-SL-messy
	0.75	0.86	0.88	122.4	844.2	769.2
XCS	-	8.15E-07	1.33E-07	-	6.19E-36	3.82E-33
XCS-SL	-	-	9.69E-03	-	-	1.15E-05

Table 1 shows the classification accuracies and populations sizes of all LCSs and p -values (for classification accuracies and for population sizes) which are calculated using the Two-tailed paired Student t-test. The population size of

XCS is quite smaller than other LCSs but the classification accuracies of XCS-SL and XCS-SL-messy are better than XCS and the positive significant differences for the classification accuracy are noted ($p < 0.01$). XCS-SL-messy improves on the classification accuracy of XCS-SL, and the positive significant difference for the classification accuracy between both systems is noted ($p < 0.01$). Additionally, XCS-SL-messy had a smaller population size than XCS-SL and the positive significant difference for the population size is noted ($p < 0.01$).

7 Conclusion

We introduced XCS-SL with a novel messy coding for memories and a novel evolutionary mechanism to find how many and where previous inputs are needed to disambiguate the current input. On the Layered Multiplexer Problem we found messy coding results in a smaller population size and does not require as high a population size limit. However, messy coding requires more training with a high population size limit than the original interval coding. On a real world sequence labeling task messy coding had higher accuracy and smaller population size than the original interval coding. These results suggest that the messy-coding in XCS-SL, combined with our new evolutionary mechanism can successfully learn accurate and compact conditions. We will evaluate other memory-using LCS on sequence labeling tasks. Finally, we will compare XCS-SL with non-evolutionary sequence labeling algorithms on a range of datasets.

References

1. Butz, M.V., Wilson, S.W.: An Algorithmic Description of XCS. *Journal of Soft Computing* 6(3-4), 144–153 (2002)
2. Kaluža, B., Mirchevska, V., Dovgan, E., Luštrek, M., Gams, M.: An Agent-based Approach to Care in Independent Living. In: de Ruyter, B., Wichert, R., Keyson, D.V., Markopoulos, P., Streitz, N., Divitini, M., Georgantas, N., Mana Gomez, A. (eds.) *AmI 2010. LNCS*, vol. 6439, pp. 177–186. Springer, Heidelberg (2010)
3. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *ICML 2001*, pp. 282–289 (2001)
4. Lanzi, P.L., Wilson, S.W.: Toward Optimal Classifier System Performance in Non-Markov Environments. *Evolutionary Computation* 8(4), 393–418 (2000)
5. Lanzi, P.L.: Extending the Representation of Classifier Conditions Part I: From Binary to Messy Coding. In: *GECCO 1999*, pp. 337–344. Morgan Kaufmann (1999)
6. Nakata, M., Kovacs, T., Takadama, K.: A Modified XCS Classifier System for Sequence Labeling. In: *Proc. of GECCO 2014*, pp. 565–572. ACM (2014)
7. Ordóñez, F.J., de Toledo, P., Sanchis, A.: Activity Recognition Using Hybrid Generative/Discriminative Models on Home Environments Using Binary Sensors. *Sensors* 13(5), 5460–5477 (2013)
8. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: *International Conf. on New Methods in Language Processing*, pp. 44–49 (1994)
9. Tomlinson, A., Bull, L.: An Accuracy Based Corporate Classifier System. *Soft Computing* 6(3-4), 200–215 (2002)
10. Wilson, S.W.: Classifier Fitness Based on Accuracy. *Evolutionary Computation* 3(2), 149–175 (1995)

Reevaluating Exponential Crossover in Differential Evolution

Ryoji Tanabe and Alex Fukunaga

Graduate School of Arts and Sciences, The University of Tokyo, Japan

Abstract. Exponential crossover in Differential Evolution (DE), which is similar to 1-point crossover in genetic algorithms, continues to be used today as a default crossover operator for DE. We demonstrate that exponential crossover exploits an unnatural feature of some widely used synthetic benchmarks such as the Rosenbrock function – dependencies between adjacent variables. We show that for standard DE as well as state-of-the-art adaptive DE, exponential crossover performs quite poorly on benchmarks without this artificial feature. We also show that shuffled exponential crossover, which removes this kind of search bias, significantly outperforms exponential crossover.

1 Introduction

Differential Evolution (DE) is an Evolutionary Algorithm (EA) that was primarily designed for real parameter optimization problems [16], and has been applied to many practical problems [14]. A DE population is represented as a set of real parameter vectors $\mathbf{x}_i = (x_1, \dots, x_D)$, $i = 1, \dots, N$, where D is the dimensionality of the target problem, and N is the population size. In each generation t , a mutant vector $\mathbf{v}_{i,t}$ is generated from an existing population member $\mathbf{x}_{i,t}$ by applying some mutation strategy. Then, the mutant vector $\mathbf{v}_{i,t}$ is crossed with the parent $\mathbf{x}_{i,t}$ in order to generate trial vector $\mathbf{u}_{i,t}$. After all of the trial vectors $\mathbf{u}_{i,t}$, $0 \leq i \leq N$ have been generated, each individual $\mathbf{x}_{i,t}$ is compared with its corresponding trial vector $\mathbf{u}_{i,t}$, keeping the better vector in the population.

Algorithm 1. exponential crossover

```
1  $\mathbf{u}_{i,t} = \mathbf{x}_{i,t}$ ,  $j$  is randomly selected from  $[1, D]$ ,  $L = 1$ ;  
2 repeat  
3   |  $u_{j,i,t} = v_{j,i,t}$ ,  $j = (j + 1)$  modulo  $D$ ,  $L = L + 1$ ;  
4 until  $\text{rand}[0, 1) < CR$  and  $L < D$ ;
```

The two most common type of crossover in DE are binomial crossover, analogous to uniform crossover in GA's, and exponential crossover, analogous to 1 or 2 point crossover in GA's [16]. Binomial crossover is implemented as follows: For each j ($j = 1, \dots, D$), if $\text{rand}[0, 1) \leq CR$ or $j = j_{rand}$, $u_{j,i,t} = v_{j,i,t}$.

Otherwise, $u_{j,i,t} = x_{j,i,t}$, where $\text{rand}[0, 1)$ denotes a uniformly selected random number from $[0, 1)$, and j_{rand} is a decision variable index which is uniformly randomly selected from $[1, D]$. $CR \in [0, 1]$ is the crossover rate. Exponential crossover is implemented as shown in Algorithm 1. The choice of crossover type determines the distribution of the number of variables that are inherited by children (trial vectors in DE terminology), as well as the contiguousness of the inherited variables. Although binomial crossover appears to be more frequently used in state-of-the-art DEs [2, 18, 23], a number of recent papers have reported successful usage of exponential crossover [3, 7, 9, 13, 24, 25].

This paper questions the continued usage of exponential crossover in DE. It has been long known in the GA community that traditional 1-point/2-point crossover introduces significant positional biases – interactions between genes that are positionally far from each other in a genome tend to be disrupted, while interactions between genes that are close to each other tend not to be disrupted [4]. This positional bias tends to result in undesirable search behavior in real-world problems, and classical 1-point/2-point crossover, while still introduced in textbooks, tends not to be used by experienced GA practitioners. Why then, is exponential crossover, which is quite similar to 1 or 2 point crossover, still used by the DE community?

We argue that exponential crossover in DE has been overrated because it successfully exploits unnatural dependencies between adjacent variables in widely used synthetic benchmarks. We show that if the benchmarks are altered to eliminate these unnatural dependencies, then exponential crossover performs very poorly. We also evaluate shuffled exponential crossover (SEC), a method for implementing exponential crossover without relying on arbitrary dependencies between adjacent variables, which was briefly suggested in [14] but to our knowledge has never been evaluated. Although exponential crossover has been one of the recommended crossover methods since the introduction of DE in 1995 [16], we believe that the use of exponential crossover needs to be carefully reconsidered in light of our experimental results.

2 Adjacent Functions: A Common But Unnatural Class of Benchmarks

In black-box optimization, synthetic benchmarks (e.g., the 13 classical functions [22], CEC benchmarks [17, 20], GECCO BBOB [5], and SOCO benchmarks [8]) are often used by EA researchers as proxies for performance on real-world problem instances. Although synthetic benchmark suites are designed in order to include representatives of many class of real-world problems (e.g. unimodal/multimodal/separable/nonseparable), previous work has pointed out that benchmark suites can have some pitfalls in using synthetic benchmarks to evaluate EA's [11, 15, 20, 21]. One specific issue is the presence of exploitable problem characteristics that do not arise in real-world problems [11].

One such “exploitable problem characteristic” found in some widely used, nonseparable synthetic benchmarks is *unnatural dependencies between adjacent*

Table 1. Nonseparable Benchmark Functions

Name	Definitions	Search Range	Properties
Rosenbrock	$f(\mathbf{z}) = \sum_{i=1}^{D-1} (100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2)$	$[-30, 30]^D$	Multimodal
Schwefels 1.2	$f(\mathbf{z}) = \sum_{i=1}^D (\sum_{j=1}^i z_j)^2$	$[-100, 100]^D$	Unimodal
Block-rotated Ellipsoid [1]	$f(\mathbf{z}) = \sum_{i=1}^{D-1} \sum_{j=1}^2 (\alpha^{j-1} (\mathbf{R}_i \cdot (z_i, z_{i+1})))^2$	$[-5, 5]^D$	Unimodal

variables – variables interacting (exclusively) with other variables that happen to have similar variable indices. For example, in the Rosenbrock function, a canonical, nonseparable function that has been widely used as an EA benchmark, each term in the summation depends on adjacent variables z_i and z_{i+1} (Table 1). However, there is no particular reason that adjacent variables should have such dependencies in real-world, black-box optimization problems, and such dependencies are an artifact of synthetic benchmarks.¹

This unnatural problem structure can be easily eliminated using the randomization procedure described in [20]. First, the permutation vector \mathbf{P} ($\mathbf{P} = P_1, \dots, P_D$) is initialized to a random permutation at the beginning of the DE run. During the DE run, whenever a trial vector \mathbf{x} is evaluated, we permute \mathbf{x} using \mathbf{P} , resulting in the permuted trial vector $\mathbf{x}' = (x_{P_1}, \dots, x_{P_D})$. Then, \mathbf{x}' is evaluated using the evaluation function, and the result is the fitness score for trial vector \mathbf{x} . This permutation effectively eliminates arbitrary dependencies between variables with consecutive in the trial vector \mathbf{x} – the dependencies are now between variables with indices that are consecutive in \mathbf{x}' , but exponential crossover, which operates on \mathbf{x} , can not exploit the consecutiveness in \mathbf{x}' .

2.1 Exponential Crossover on Adjacent/Distributed Functions

We evaluate exponential crossover on (1) functions with dependencies between lexicographically adjacent variables, i.e., standard versions of widely used synthetic benchmarks, and (2) modified versions of functions in (1) where the permutation method described above is used to randomize the variable dependencies and eliminate the adjacent dependency structure. Following [4], we call functions of the former class *adjacent functions*, and functions of the latter class *distributed functions*.

We used the 3 nonseparable, adjacent functions in Table 1. The Rosenbrock and Schwefels 1.2 functions are well-known, classical benchmark functions that have been widely used to evaluate EA's [22]. The Block-rotated Ellipsoid [1] is a partially separable function designed to only have dependencies between

¹ Of course, there are real-world problems that can be represented in such a way as to have dependencies between adjacent variables [1] – we are merely arguing that these are not representative of *black-box* optimization problems.

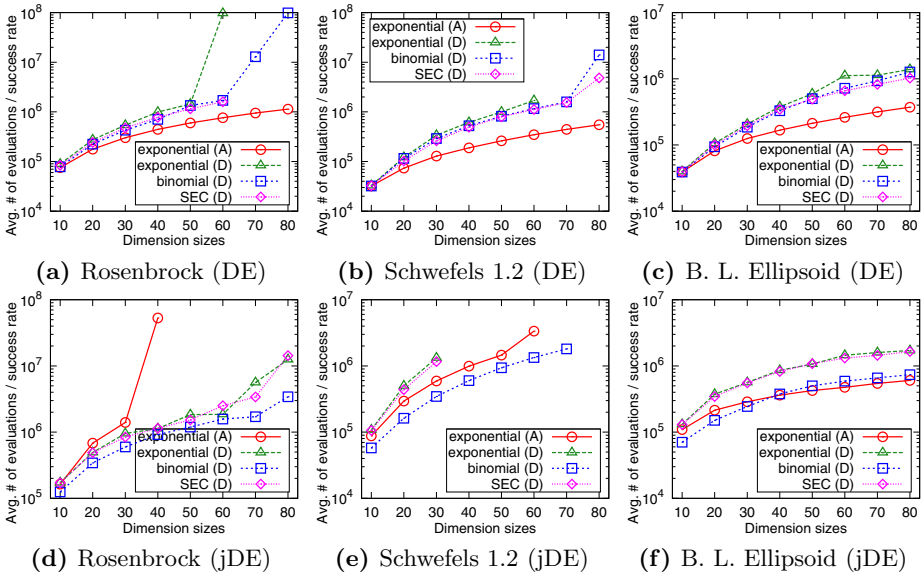


Fig. 1. Evaluation of various crossover operators (standard DE and jDE). (A) and (D) stand "Adjacent" and "Distributed" respectively. The horizontal axis represents the dimensionality D , and the vertical axis represents the average fitness evaluations (for successful runs) divided by success rate.

z_i and z_{i+1} . Here, $\mathbf{z} = (y_{P_1}, \dots, y_{P_D})$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$, and for each function, the location of the global optimum has been shifted by offset \mathbf{o} ($\mathbf{o} = o_1, \dots, o_D$), where each component of \mathbf{o} is a uniformly generated random offset. For adjacent functions, the permutation vector $\mathbf{P} = (1, \dots, D)$, and for distributed functions, \mathbf{P} is a randomly generated ordering such as $\mathbf{P} = (6, 1, 3, \dots)$. The 2×2 rotation matrix \mathbf{R}_i is uniformly generated according to the method of [15] and $\alpha = 1e+6$.

We studied problems with 10 – 80 dimensions. Each DE run continues until either (1) the difference between the best-so-far and the optimal solution $\leq 1e-8$, in which case we treat the run as a "success", or (2) the # of objective function calls exceeds 2.0×10^6 , in which case the run is treated as a "failure". On each problem, each algorithm is executed 50 times. Following [6], our evaluation metric is the average # of fitness evaluations in successful runs divided by the # of successes. Small values of this metric indicate a fast and stable search.

We use standard DE [16], as well as the state-of-the-art adaptive DE variants jDE [2], JADE [23], and SHADE [18].² The standard DE used a population size of 100 and $F = 0.5$, and the most commonly used, and rand/1 mutation strategy – this is a standard configuration in the DE literature [2, 23]. In addition to exponential crossover, we also ran the experiments with binomial

² jDE [2], JADE [23], SHADE [18] were originally designed to use binomial crossover; in order to evaluate exponential crossover on state-of-the-art DE's, we modified these to use exponential crossover.

crossover for comparison. On the standard DE, for each crossover method, for each benchmark function, and for each dimensionality (D), we use the value of $CR \in \{0.90, 0.91, \dots, 0.99\}$ that yields the best performance according to the performance metric defined above. For jDE, JADE, and SHADE, which automatically adjusts CR , we use the control parameters recommended in the original papers on these adaptive methods [2, 18, 23].

The results for standard DE and jDE are shown in Figure 1. Detailed results for JADE and SHADE are in the supplemental material [19] due to space constraints, but the SHADE and JADE results are qualitatively similar to the jDE results. “Shuffled exponential crossover (SEC)” is explained in Section 3. For cases where all runs failed (success rate for 50 trials = 0), then the data is not shown. Since binomial crossover behaves almost identically for adjacent and distributed functions, only the distributed function results are shown.

Figure 1 shows that exponential crossover performs much better on adjacent functions compared to distributed functions. The performance gap increase as the dimensionality increases. For standard DE, exponential crossover outperforms binomial crossover on all adjacent functions, for all dimensionalities. In stark contrast, on the distributed functions, the performance of exponential crossover drops significantly – for all the distributed functions, for all D , exponential crossover performs worse than binomial crossover. In particular, on the distributed-Rosenbrock and distributed-Schwefels 1.2 functions, for $D \geq 70$ dimensions, exponential crossover fails on every single run. The results are similar for jDE. Aside from the results on the Rosenbrock function (Figure 1(d)), the performance of exponential crossover on distributed functions is clearly worse than on the adjacent functions.

These results clearly show that the performance of exponential crossover on nonseparable function benchmarks such as Rosenbrock and Schwefels 1.2 depends on an arbitrary feature of these synthetic benchmarks – variable dependencies between adjacent variables. Given essentially the same, nonseparable functions without this arbitrary structure, exponential crossover performs much worse (significantly worse than binomial crossover). Functions such as Rosenbrock and Schwefels 1.2 have been part of benchmark suites used by to evaluate DE since the original paper introducing DE [16], and continue to be used today [3, 7, 9, 13, 24, 25]. As a consequence, exponential crossover has been inaccurately overrated as a DE crossover operator for black-box optimization.

How fragile is exponential crossover to perturbations in the variable index order? Instead of completely randomizing the variable index order, we investigate the effect of gradually decreasing the dependency between adjacent variable indices by applying $n = 1, \dots, D$ randomized swaps to the variable indices. As n increases, the number of dependencies between adjacent variables decreases. We ran standard DE on 50-dimensional problems (same setup as in the previous experiment). For each problem, we used the CR value that performed best for each n . Figure 2 shows that for all of the functions, exponential crossover performs best when $n = 0$, and rapidly degrades as n increases, i.e., exponential

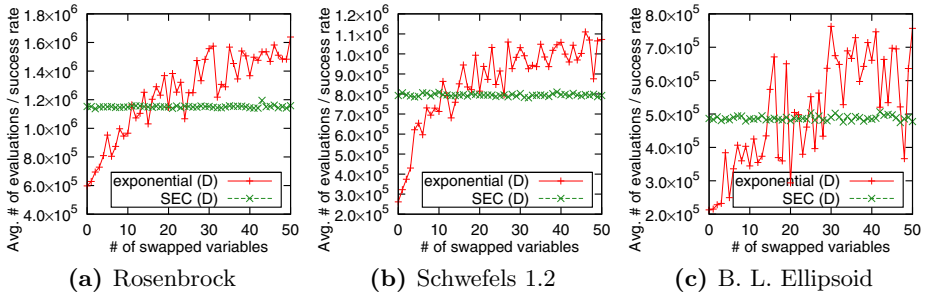


Fig. 2. Effect of gradually decreasing dependencies between adjacent variables for standard DE using exponential crossover and SEC on 50-dimensional problems. Horizontal axis = # of randomized swaps ($n = 1, \dots, D$); Vertical axis = average fitness evaluations (for successful runs) divided by success rate.

crossover is quite fragile with respect to perturbations in dependencies between adjacent variables.

3 Shuffled Exponential Crossover

The experiments in Section 2 showed that exponential crossover performs poorly on distributed functions. To alleviate this problem, we evaluate shuffled exponential crossover. Caruana et al observed that 1-point crossover in GA’s introduced a very strong search bias in that variables located close to each other in a linear genome (i.e., variable indices with similar lexicographic values) tended to be inherited by the same child, while variables located far apart tend to be inherited by different children [4]. To eliminate this bias, they proposed shuffle crossover [4]: First, the variable indices of the parents are randomly shuffled (the same shuffle is applied to both parents). Next, standard 1-point crossover is applied to the shuffled genomes. Finally, the indices are restored to their pre-shuffled states.³

Price et al note that exponential crossover in DE is subject to the same bias as 1-point crossover in GA’s, and suggested that the shuffling mechanism from shuffle crossover can be added to exponential crossover in order to alleviate this problem [14]. In this paper, we call this *shuffled exponential crossover* (SEC). The algorithm is shown in Algorithm 2. Recently, Lin et al evaluated a mechanism called non-consecutive exponential crossover which is in fact, the same as SEC [12]. However, they do not analyze the effect of this mechanism in the context of dependencies among variables as we do. In fact, [12] claims “in non-separable ridge functions (Rosenbrock and Schwefels 1.2 function), differential evolution algorithms with consecutive crossover are more reliable than those with non-consecutive crossover” (this is clearly contradicted by our result above).

³ While shuffle crossover is similar to uniform crossover in that the genes that are exchanged are dispersed throughout the genome, the # of genomes that are exchanged has a very different probability distribution, and there is a different search bias.

Algorithm 2. Shuffled Exponential Crossover (SEC)

```

1  $\mathbf{u}_{i,t} = \mathbf{x}_{i,t}$ ,  $k = 1$ ,  $\mathbf{S}(= s_1, \dots, s_D)$  is randomly shuffled permutation  $\{1, \dots, D\}$ ;
2 repeat
3   |  $j = s_k$ ,  $u_{j,i,t} = v_{j,i,t}$ ,  $k = k + 1$ ;
4 until  $\text{rand}[0, 1) < CR$  and  $k < D$ ;
```

Table 2. # of Adjacent Functions in Standard Benchmark Suites

Benchmarks	# of adjacent functions	Function
13 classical [22]	2 / 13	F_3, F_5
CEC 2005 [17]	4 / 25	F_2, F_4, F_6, F_{13}
CEC 2010 [20]	2 / 20	F_{19}, F_{20}
GECCO BBOB [5]	1 / 24	F_8
SOCO benchmarks [8]	10 / 19	$F_3, F_8, F_9, F_{11}, F_{12}, F_{13}, F_{14}, F_{16}, F_{17}, F_{18}$

Figure 1 shows the results of SEC for standard DE as well as jDE for the same problems/settings as in Section 2.1.⁴ Unlike exponential crossover (but, similar to binomial crossover), the performance of SEC is unaffected by whether the test function is adjacent or distributed; thus, only the distributed function results are shown. Figure 2 also shows the results for SEC with best CR value when $n = 0$ for the same problems/settings as in Section 2.1. The performance of SEC is clearly shown to be independent of n .

Figure 1 shows that overall, SEC slightly outperforms exponential crossover on the distributed functions. However, on the distributed Rosenbrock function (60 dimensions), SEC significantly outperforms exponential crossover. Also, on the distributed-Schwefels 1.2 function, SEC is competitive with binomial crossover, even on $70 \geq$ dimensional problems where exponential crossover failed on every single run. These results show that while exponential crossover depends on the ordering of variables (i.e., whether the function is adjacent or distributed), SEC, as expected, does not depend on the index positions of the variables and yields a much more stable search performance as a result.

4 Is Exponential Crossover Overrated?

Past evaluations of DE crossover operators on standard benchmark test suites need to be reconsidered in light of our analysis of the interaction between adjacent functions and exponential crossover. Functions with dependencies between adjacent variables (e.g., Rosenbrock, Schaffer F_7 , Whitley's composite functions [21]) are included in widely used benchmark suites. Table 2 shows the

⁴ Results for JADE and SHADE are in the supplemental material [19].

Table 3. SEC vs. binomial and exponential crossover on the CEC2014 benchmark functions [10] for $D = 50$ (Wilcoxon rank-sum test significance threshold $p < 0.05$)

		exponential				binomial			
		DE	jDE	JADE	SHADE	DE	jDE	JADE	SHADE
vs. SEC	# of better	1	0	0	0	12	18	19	13
Wilcoxon rank-sum	# of worse	8	2	3	5	9	5	7	6
(significance: $p < 0.05$)	# of no sig.	21	28	27	25	9	7	4	11

number of adjacent functions out of the total number of functions in the typical benchmarks. The larger the number of adjacent functions in the benchmark suite, the more favorable the suite is for methods that exploit the adjacent structure, such as DE with exponential crossover. In particular, note that 10 out of 19 functions in the recent Soft Computing Journal (SOCO) benchmarks [8] are adjacent functions, making it particularly vulnerable to exploitation by exponential crossover. *In fact, all 7 of the DE algorithms submitted to the SOCO special issue evaluation used exponential crossover.* It would seem that due to the presence of adjacent functions, exponential crossover has been overrated in previous evaluations (assuming that benchmark suites are supposed to model true “black-box” scenarios where there is no a priori reason to believe that adjacent variables have dependencies).

How do DE crossover operators compare on a benchmark set that does not contain any adjacent functions? The recently proposed CEC2014 benchmark set [10] consisting of 30 problems, does not include any adjacent functions. We evaluated SEC vs exponential crossover vs binomial crossover on DE, jDE, JADE, and SHADE on the CEC2014 benchmarks (in 10, 30, and 50 dimensions), following the evaluation methodology specified in the CEC2014 benchmark competition [10]. The overall results for $D = 50$ dimensions are shown in Table 3.⁵ As shown in Table 3, for a diverse benchmark set, SEC outperforms exponential crossover for all DE algorithms. Binomial crossover performs best for all DE algorithms.

5 Conclusion

This paper showed that exponential crossover, one of the standard crossover methods in DE, implicitly exploits an unnatural structure found in some synthetic benchmark problems, including some widely used, nonseparable functions (Rosenbrock and Schwefels 1.2), where there are strong dependencies between variables with consecutive indices. We showed that exponential crossover performs significantly worse if we slightly perturbing these classical benchmarks to remove these arbitrary, lexicographic dependencies, i.e., after artificial dependencies between adjacent variables are removed, the “true” performance of

⁵ The results for $D = 10, 30$ are in the supplemental data [19].

exponential crossover appears to be significantly worse than previously believed. We believe that for synthetic benchmarks [5, 8, 17, 20, 22], the performance of exponential crossover has been overrated [3, 7, 9, 13, 24, 25]. Thus, the suitability of exponential crossover should be reevaluated in light of our results. We showed that SEC, which does not implicitly assume sequential dependencies between variables and does not have the same search bias as exponential crossover, significantly outperforms exponential crossover overall, and is competitive with binomial crossover. While SEC was suggested by [14], and also proposed by [12], our work is the first to identify the specific weaknesses described above for exponential crossover and show that shuffling results in improved overall performance. Although there is still no clear criteria to determine whether binomial crossover or SEC should be used for a particular problem (a direction for future work), we believe that we have presented sufficient evidence to suggest that plain, exponential crossover should no longer be considered as an appropriate, default crossover operator for DE. SEC (or binomial crossover) should be used instead of exponential crossover, unless there is some prior knowledge that there are dependencies between consecutive variables.

As we showed for exponential crossover in DE, algorithms that (intentionally or unintentionally) exploit this dependency can appear to perform much better than they would actually perform on real-world problems without this artificial structure. As shown in Section 2, randomizing the lexicographic positions of the variables for all benchmark functions, as suggested by [20] is a simple method for avoiding this benchmarking pitfall, and we believe that black-box benchmark suites should apply this randomization to avoid unintentionally biasing the evaluation results.

References

1. Bouzarkouna, Z., Auger, A., Ding, D.Y.: Local-meta-model CMA-ES for partially separable functions. In: GECCO, pp. 869–876 (2011)
2. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Tran. Evol. Comput.* 10(6), 646–657 (2006)
3. Brest, J., Maučec, M.S.: Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Comput.* 15(11), 2157–2174 (2011)
4. Caruana, R., Eshelman, L.J., Schaffer, J.D.: Representation and Hidden Bias II: Eliminating Defining Length Bias in Genetic Search via Shuffle Crossover. In: *IJ-CAI*, pp. 750–755 (1989)
5. Hansen, N.: GECCO BBOB (2014), <http://coco.gforge.inria.fr/doku.php>
6. Hansen, N., Kern, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In: Yao, X., et al. (eds.) *PPSN VIII*. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004)
7. Herrera, F., Lozano, M., Molina, D.: Components and parameters of *de*, *real-coded chc*, and *g-cmaes*. Technical report, Univ. of Granada (2010)
8. Herrera, F., Lozano, M., Molina, D.: Test suite for the spec. iss. of *Soft Computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems*. Technical report, Univ. of Granada (2010)

9. LaTorre, A., Muelas, S., Peña, J.M.: A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: A scalability test. *Soft Comput.* 15(11), 2187–2199 (2011)
10. Liang, J.J., Qu, B.Y., Suganthan, P.N.: Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical report, Zhengzhou Univ. and Nanyang Technological Univ. (2013)
11. Liang, J.J., Suganthan, P.N., Deb, K.: Novel Composition Test Functions for Numerical Global Optimization. In: *Swarm Intell. Symp.*, pp. 68–75 (2005)
12. Lin, C., Qing, A., Feng, Q.: A comparative study of crossover in differential evolution. *J. Heuristics* 17(6), 675–703 (2011)
13. Noman, N., Iba, H.: Accelerating Differential Evolution Using an Adaptive Local Search. *IEEE Tran. Evol. Comput.* 12(1), 107–125 (2008)
14. Price, K.V., Storn, R.N., Lampinen, J.A.: *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer (2005)
15. Salomon, R.: Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems* 39(3), 263–278 (1996)
16. Storn, R., Price, K.: *Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces*. Technical report, International Computer Science Institute, Berkeley, CA (1995)
17. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological Univ. (2005)
18. Tanabe, R., Fukunaga, A.: Success-History Based Parameter Adaptation for Differential Evolution. In: *IEEE CEC*, pp. 71–78 (2013)
19. Tanabe, R., Fukunaga, A.: Supplemental material (2014), <https://sites.google.com/site/tanaberyoji/home/ppsn2014-supplement.pdf>
20. Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T.: Benchmark Functions for the CEC 2010 Special Session and Competition on Large-Scale Global Optimization. Technical report, Univ. of Science and Technology of China (2010)
21. Whitley, D., Mathias, K., Rana, S., Dzubera, J.: Evaluating evolutionary algorithms. *Artificial Intelligence* 85, 245–276 (1996)
22. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming Made Faster. *IEEE Tran. Evol. Comput.* 3(2), 82–102 (1999)
23. Zhang, J., Sanderson, A.C.: JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Tran. Evol. Comput.* 13(5), 945–958 (2009)
24. Zhao, S., Suganthan, P.N.: Empirical investigations into the exponential crossover of differential evolutions. *Swarm and Evol. Comput.* 9, 27–36 (2013)
25. Zhao, S., Suganthan, P.N., Das, S.: Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Comput.* 15(11), 2175–2185 (2011)

An Extended Michigan-Style Learning Classifier System for Flexible Supervised Learning, Classification, and Data Mining

Ryan J. Urbanowicz, Gediminas Bertasius, and Jason H. Moore

Institute for Quantitative Biomedical Sciences, Department of Genetics
Geisel School of Medicine, Lebanon, NH, USA
{ryan.j.urbanowicz,jason.h.moore}@dartmouth.edu
<http://www.epistasis.org/>

Abstract. Advancements in learning classifier system (LCS) algorithms have highlighted their unique potential for tackling complex, noisy problems, as found in bioinformatics. Ongoing research in this domain must address the challenges of modeling complex patterns of association, systems biology (i.e. the integration of different data types to achieve a more holistic perspective), and ‘big data’ (i.e. scalability in large-scale analysis). With this in mind, we introduce ExSTraCS (Extended Supervised Tracking and Classifying System), as a promising platform to address these challenges using supervised learning and a Michigan-Style LCS architecture. ExSTraCS integrates several successful LCS advancements including attribute tracking/feedback, expert knowledge covering (with four built-in attribute weighting algorithms), a flexible and efficient rule representation (handling datasets with both discrete and continuous attributes), and rapid non-destructive rule compaction. A few novel mechanisms, such as adaptive data management, have been included to enhance ease of use, flexibility, performance, and provide groundwork for ongoing development.

Keywords: Learning Classifier System, Genetics, Epidemiology, Epistasis, Heterogeneity, Evolutionary Algorithm, Systems Biology.

1 Introduction

Machine learning algorithms driven by evolutionary mechanisms offer a promising avenue for data mining within complex, noisy problem domains. Michigan-style learning classifier systems (LCS) constitute a unique class of algorithms that distribute learned patterns over a collaborative population of individually interpretable (IF:THEN) rules, allowing them to flexibly and effectively describe complex and diverse problem spaces found in behavior modeling, function approximation, classification, and data mining. Michigan LCS algorithms apply iterative rather than batch-wise learning, meaning that rules are evaluated and evolved one data instance at a time. This makes them naturally well-suited to learning different problem niches found in multi-class, latent-class, or heterogeneous problem domains. LCS algorithms are fundamentally multi-objective,

evolving rules toward maximal accuracy and generality (i.e. rule simplicity) to improve predictive performance [1]. We focus on classification and data mining problems in genetics and epidemiology where risk factors that explain variation in disease phenotypes are sought. Certain complicating phenomena are known to interfere with the traditional mapping of genotype to phenotype [2]. We explicitly consider two such phenomenon; epistasis (i.e. gene-gene interaction) and genetic heterogeneity. Also, new systems biology approaches will require the integration of different data types (e.g. genetic, epigenetic and environmental) embracing a more holistic perspective when searching for predictive or causal disease risk factors. Difficulty is further compounded in the context of large-scale bioinformatic investigations where the computational and methodological limitations hinder scalability with increasing numbers of attributes and/or training instances.

With these challenges in mind, we introduce the Extended Supervised Tracking and Classifying System (ExSTraCS). ExSTraCS has been primarily designed to address complex, noisy, supervised learning, single-step problem domains with 2 or more balanced/imbalanced classes, and with continuous or discrete attributes. ExSTraCS is descended from a lineage of Michigan-style LCS algorithms, founded on the architecture of Wilson's Extended Classifier System (XCS) [3], the most successful and best-studied LCS algorithm to date. The Supervised Classifier System (UCS) [4] replaced XCS's reinforcement learning scheme with a supervised learning strategy to deal explicitly with single-step problems such as classification and data mining. Comparing select Michigan and Pittsburgh-style LCS algorithms, UCS showed particular promise when applied to complex biomedical data mining problems with patterns of epistasis and heterogeneity [5,6]. UCS inspired two algorithmic expansions named Attribute Tracking and Feedback UCS (AF-UCS) and Expert Knowledge UCS (UCS-EK). AF-UCS introduced mechanisms that improved learning and uniquely allowed for the explicit characterization of heterogeneous patterns and the identification of candidate disease subgroups [7,8]. UCS-EK incorporated expert knowledge into UCS learning for smart population initialization, directed rule discovery, and reduced run time [9]. Recently, novel rapid rule compaction strategies were developed and evaluated for post-processing rule populations to enhance interpretability and improve predictive performance [10]. ExSTraCS merges successful components of this algorithmic lineage with other valuable LCS research, and a redesigned UCS-like framework with a few novel features. In addition to integrating attribute tracking/feedback, expert knowledge covering, and rapid rule compaction, ExSTraCS (1) adopts a flexible and efficient rule representation similar to the one described in [11], to accommodate data with both discrete and continuous attributes, (2) outputs attribute tracking scores and global statistics (in addition to a rule population) for significance testing, and visualization-guided knowledge discovery as described in [12], (3) includes an adaptive data detection scheme to adjust the algorithm to the characteristics of the dataset, and (4) includes a built-in selection of four attribute weighting algorithms to discover potentially useful expert knowledge as a pre-processing step.

2 ExSTraCS

The Extended Supervised Tracking and Classifying System (ExSTraCS) combines a number of existing and completely novel aspects into a single, flexible LCS framework aimed at overall functionality, ease of use, as a platform for ongoing algorithmic development. The algorithm itself is coded in Python, well annotated, and freely available on *sourceforge.net*. We begin with an overview of ExsTraCS referencing a schematic of major components given in Figure 1. We follow with details of components that differentiate ExSTraCS from the UCS or XCS algorithms.

ExSTraCS begins with (A) data pre-processing, followed by (B) algorithm learning/training, and ending with (C) rule population post-processing. (A) ExS-TraCS will accept a finite dataset with some number of independent attributes and a single class variable as the training dataset. A testing dataset may be optionally loaded for complete rule population evaluations. Adaptive data management initially determines and stores key characteristics of this dataset for use during learning iterations. Lastly, expert knowledge (EK) may be loaded or discovered from the dataset using one of four implemented attribute weighting algorithms. These weights are converted describing relative probabilities that attributes in the data will be valuable for discriminating class/endpoint. (B) The core ExSTraCS algorithm largely follows a typical iterative Michigan-style

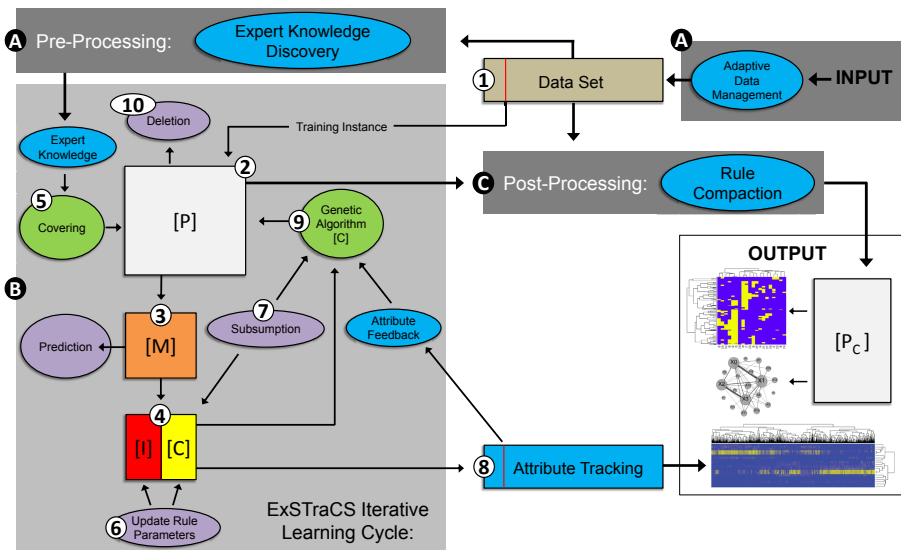


Fig. 1. ExSTraCS Schematic: Ovals are mechanisms, bordered squares are sets of either data or classifiers, green = classifier discovery mechanism, purple = traditional LCS mechanism, and blue = mechanisms unique to ExSTraCS

learning cycle that includes the following 10 steps repeatedly up to a maximum number of learning iterations: (1) One training instance is taken from the dataset without replacement. (2) The training instance is passed to a population [P] of classifiers/rules that is initially empty. A *classifier* is a simple IF/THEN rule comprised of a condition (i.e. specified attribute states), and what is traditionally referred to as an action (i.e. the state of the class or endpoint). (3) A match set [M] is formed, that includes any classifier in [P] that has a condition matching the training instance. (4) [M] is divided into a correct set [C] and an incorrect set [I] based on whether each classifier specified the correct or incorrect class/phenotype. (5) If, after steps 3 and 4, [C] is empty, covering applies expert knowledge to intelligently generate a matching and 'correct' classifier added to [M] and [C]. (6) For every classifier in [P], a number of parameters are maintained and updated throughout the learning process such as: numerosity (the number of copies of a given classifier in [P]), rule accuracy which is the proportion of times in which a classifier has been in a [C] over all times it has been in a [M]; and classifier fitness, which is simply equal to classifier accuracy in this implementation (i.e. $\nu = 1$). We had previously observed that placing too much emphasis on optimal accuracy in calculating fitness led to dramatic overfitting in noisy problem domains [5]. Classifier parameters (e.g. fitness) are updated for classifiers within [C] and [I]. (7) Subsumption, a generalization mechanism is applied to [C] [3]. A similar subsumption mechanism is also applied to new classifiers generated by the genetic algorithm (GA). (8) Classifiers in [C] are used to update attribute tracking scores for the current training instance. (9) The GA uses tournament selection to pick two parent classifiers from [C] based on fitness and generates two offspring classifiers which are added to [P]. The GA includes two discover operators: crossover and mutation ($\chi = 0.8$ and $\nu = 0.04$, respectively). All classifiers in [C] and [I] are returned to [P]. (10) Lastly, whenever the size of [P] is greater than the specified maximum, a deletion mechanism decrements the numerosity of a classifier (assuming it is > 1) or removes it from [P]. Deletion probability is a function of classifier numerosity, average [M] size and is inversely proportional to fitness. Notably, ExSTraCS cycles do not alternate between an explore/exploit phases as described in XCS [3] due to supervised learning. However, for performance tracking and prediction evaluation, a prediction array is generated every iteration from [M] to obtain a class prediction. A class prediction is made by a fitness weighted vote of all classifiers within [M]. The class with the largest 'vote' is the predicted class. (C) After all learning iterations have completed, rule compaction is applied as a post-processing step to remove poor and/or redundant rules from [P] to yield [P_c]. Upon request, ExSTraCS will yield up to four distinct output files after the final iteration, or any iteration at which a full evaluation is requested. These include (1) the population of classifiers collectively constituting the prediction 'model' (Note: ExSTraCS is uniquely set up to load a given population file and continue learning from where it left off), (2) population statistics, summarizing major performance statistics including global training and testing accuracy of the classifier population [12], (3) co-occurrence scores for the top specified pairs of attributes in the dataset

Quaternary Knowledge Representation		Mixed Discrete-Continuous Attribute-List Knowledge Representation	
Rule Condition:	[#, 2, #, #, 0, #, 1, 2, #, #]	Attribute Reference:	[1, 4, 6, 7]
Classification/Action:	1	Rule Condition:	[2, 0, [0.4 - 0.7], 'high']
		Classification/Action:	1

Fig. 2. Knowledge Representations: Quaternary vs. Mixed Discrete-Continuous Attribute List. The ‘#’ symbol indicates ‘attribute not specified’, which matches any attribute state. The mixed representation only stores specified attributes, represents continuous value states as flexible ‘ranges’, and allows for non-numerical states.

[12], and (4) attribute tracking scores for each instance in the dataset [7]. These outputs may be evaluated and visualized to facilitate knowledge discovery as described in [12].

Adaptive Data Management. We introduce a simple adaptive data management (ADM) scheme to facilitate ease of use, improve efficiency, and algorithmic adaptation to different datasets. ADM will load and format training (and optionally testing) data, automatically identifying key characteristics including: number of attributes, number of instances, the location of the endpoint variable column, and the location of a column for instance identifiers (optional, but useful in tying attribute tracking scores back to specific individuals in the dataset). Also, ADM examines state values for all attributes and applies a user defined run parameter (*discreteAttributeLimit*) to determine and store whether each attribute is to be treated as discrete or continuous. If discrete, each possible state value stored, while if continuous, the maximum and minimum values are stored, for use in limiting covering and GA mechanisms. We plan to expand ADM to store state frequency information which we expect can be applied to further improve performance. ADM reduces redundancy, simplifies data formatting requirements, and paves the way for further algorithmic enhancements.

Knowledge Representation. Our prior implementations of UCS [5], AF-UCS [7], and UCS-EK [9] were coded with a quaternary knowledge representation to operate on single nucleotide polymorphism (SNP) case/control data. SNPs are discrete genetic attributes with encoded states (0,1,or 2). ExSTraCS adopts a mixed discrete-continuous attribute-list knowledge representation (see Figure 2) allowing learning on datasets with discrete and/or continuous attributes. This strategy is quite similar to the one proposed by Bacardit in [11] which extended the attribute-list knowledge representation (ALKR), designed for continuous attributes, with the GABIL discrete attribute representation [13]. ALKR only stores information about attributes that are specified in a classifier which significantly reduces run time in both matching and attribute tracking. This effect is particularly important in datasets with a large number of non-predictive attributes. ExSTraCS keeps the ALKR representation but avoids the GABIL

representation in favor of a simpler but less generalizable strategy for representing discrete attribute states. Specifically, classifier conditions can only specify one state for discrete attributes, while GABIL allows for some subset of attribute states to be simultaneously specified. While this may be advantageous for evolving a maximally compact rule-set, this approach is not in-line with the global approach to knowledge discovery proposed in [12] which relies on important attributes being specified more often across rules in the greater population, a valuable component to addressing significant noise in LCS data mining. Additionally, this stricter representation yields individual rules that are arguably easier to interpret (less ambiguity within the IF/THEN statement) and that are likely to be more accurate individual predictors (since they independently capture a more specific set of attribute states). This representation requires modifications to both covering and the genetic algorithm in order to handle continuous attributes (implemented as described in [11]).

Attribute Tracking and Feedback. Attribute tracking (AT) is akin to long-term memory for supervised, iterative learning (see (8) in Figure 1). For a finite training dataset, a vector of accuracy scores is maintained for each instance in the data. In other words, for every instance in the data we increase attribute weights based on which attributes are being specified in rules found in [C] every iteration. Post-training, these scores can be applied to characterize patterns of association in the dataset, in particular heterogeneous patterns which might suggest clinical patient subgroups that may be targeted for research, treatment, or preventative measures [7]. Note that using attribute tracking alone does not impact learning performance. Attribute feedback (AF) is applied to the GA mutation and crossover operators, probabilistically directing rule generalization based on the AT scores from a randomly selected instance in the dataset. The probability that AF will be used in the GA is proportional to the algorithm's progress through the specified number of learning iterations (i.e. AF is applied infrequently early-on, but frequently towards the end). Note that in developing ExSTraCS we realized that AF-UCS was not using the AT scores from the current training instance (as mistakenly described in [7]), but rather the scores from a neighboring instance. This 'error' turned out to be essential to recapitulate attribute feedback performance. AF speeds up effective learning by gradually guiding the algorithm to more intelligently explore reliable attribute patterns. These mechanisms and their application are further detailed in [7] and [8].

Expert Knowledge Covering. Previous work exploring the utilization of expert knowledge (EK) in UCS indicated that EK, utilized as probabilistic weights for specifying attributes in rules, significantly sped up learning when applied to covering, but yielded inconsistent success when applied to GA operators [9]. Therefore, ExSTraCS adopts EK covering. EK is essentially an external bias introduced to better guide learning, such that attributes more likely to be important tend to be specified more often when covering. In other words, classifiers tend to be initialized in parts of the problem space deemed by the EK to be

mostly likely to predict class status. Notably, the utility of EK is only as good as the quality of the information behind the weights. EK covering is implemented in ExSTraCS as described in [9] including the calculation of EK probability weights from raw EK scores, and the application of these weights within the covering mechanism. In theory the source of EK is up to the user (i.e. classifier population initialization can be biased towards whatever attributes desired). In [9], raw EK scores were obtained externally using a rapid attribute weighting algorithm called SURF [14], designed to estimate attribute quality, in terms of predicting class status. For convenience and flexibility, we have implemented SURF as well as three other related attribute weighting algorithms into ExSTraCS (ReliefF [15], SURF* [16], and MultiSURF [17]) from which the user may select and discover EK scores for their respective datasets. Each algorithm has been re-implemented to allow for discrete and continuous attributes. ExSTraCS handles EK discovery is a pre-processing step (see (A) in Figure 1). This study applies MultiSURF to discover EK, as it is the newest and most powerful.

Rule Compaction. ExSTraCS makes the six rule compaction strategies evaluated in [10] available to post-process the classifier population (see (C) in Figure 1). Rule compaction utilizes the whole training dataset to consolidate the classifier population with the goal of improving interpretation and knowledge discovery. Comparisons in [10] suggested that simple Quick Rule Filtering (QRF) was both the fastest, and particularly was well suited to the theme of global knowledge discovery [12] where it is more important to preserve or improve performance than to minimize rule population size (useful for knowledge discovery by manual rule inspection) [10]. This study applies QRF.

Miscellaneous. ExSTraCS naturally handle missing data points without requiring imputation bias. Missing data points require a standard unique designation, and when encountered they match any attribute state specified in the condition of a classifier. If desired, imputation can still be performed prior to running ExSTraCS, but this is not currently built-in. ExSTraCS can perform a complete evaluation of the classifier population as a whole at user specified iterations, including assessments of training and testing accuracy. Balanced accuracy is used to avoid accuracy calculation bias in multi-class and imbalanced datasets. ExSTraCS centralizes and organizes all run parameters in a readable configuration file, required to run the algorithm. Included in these parameters is the option to deactivate major ExSTraCS mechanisms as desired, such as attribute tracking/feedback, expert knowledge, and rule compaction.

3 Results and Discussion

We evaluate ExSTraCS using two separate simulation studies (with discrete or continuous attributes respectively) each including a total of 960 diverse datasets (spanning from easily solvable to currently unsolvable) with underlying predictive models that simulated patterns of epistasis and heterogeneity concurrently.

Discrete attribute SNP datasets very similar to those used in [7,9,10] were simulated using GAMETES [18] with; architectures at maximum and minimum detection difficulty, heritabilities (i.e. the proportion of class variance that can be attributed to modeled attributes) of (0.1, 0.2, or 0.4), a minor allele frequency of 0.2, 20 attributes (only four of which were predictive and 16 were noise), sample sizes of (200, 400, 800, or 1600) and a heterogeneous mix ratio of either (50:50 or 75:25) (e.g. 75% of instances were generated from one epistatic model, and 25% were generated from a different one). 20 replicates of each dataset were analyzed and 10-fold cross validation (CV) was employed to measure average testing accuracy and account for over-fitting. 960 corresponding continuous-valued versions of these datasets were generated by transforming discrete values into random values within specified continuous intervals (e.g. a discrete attribute with states 0, 1, or 2, was transformed to have a random continuous value within the respective ranges of 0-50, 50-100, or 100-150. ExSTraCS was run up to 200,000 learning iterations but performance was also evaluated after only 10,000 iterations. Pair-wise statistical comparisons were made using the Wilcoxon signed-rank tests. All statistical evaluations were completed using R.

The focus of this analysis was two-fold; (1) comparing the performance of our previous UCS-based core algorithm to the core ExSTraCS algorithm, where ‘core’ refers to the learning cycle without EK, AT/AF or rule compaction activated, and (2) comparing core ExSTraCS to performance when these separate mechanisms are activated, in order to demonstrate their combined value. These comparisons are performed using the discrete attribute simulation study summarized in Table 1 over a set of key performance metrics. ‘Both Power’ is the ability to correctly identify both two-locus heterogeneous models. ‘Single Power’ is the ability to have found at least one. ‘Co-occur. Power’ indicates the ability to detect the correct heterogeneous pattern. Generality refers to classifier generality, or the average proportion of unspecified attributes across the classifier population. Macro Population refers to the number of unique classifiers in the classifier population. Previously, we demonstrated that UCS yielded the most promising performance on these types of simulated datasets when compared to XCS, MCS, GALE and GAssist (LCS algorithms) [5,6]. Therefore we utilize the ‘core’ version of UCS used in [7,9,10] as the standard of comparison for ExSTraCS. Notice that in Table 1 the ‘core’ ExSTraCS p-values are from a comparison to ‘core’ UCS, while all other p-values correspond to comparisons between ‘core’ ExSTraCS and ExSTraCS with respective mechanisms activated. As expected, the mixed-ALKR knowledge representation added to ExSTraCS significantly and consistently reduces run time by over 30% on average, when comparing ‘core’ UCS to ‘core’ ExSTraCS. We expect this difference to be even more dramatic in datasets with > 20 attributes. Interestingly, a significant increase in testing accuracy is also observed. Next we compare ExSTraCS performance when activating major new mechanisms including (1) EK, (2) AF, (3) EK + AF, and (4) EK + AF + QRF. Performance improvements from EK and AF alone were consistent with those observed in [7,9]. Further performance improvements were observed when combining mechanisms. Additionally, comparing UCS to ExSTraCS with

Table 1. Average performance over all 960 discrete-valued datasets

10,000 Iterations (Early Performance)									
Performance Statistics	UCS	ExSTraCS			AF		EK-AF		
	Core	Core	<i>p</i>	EK	<i>p</i>	AF	<i>p</i>	EK-AF	<i>p</i>
Training Accuracy	.8569	.8640	↑**	.8628	↓**	.8635	-	.8630	↓**
Test Accuracy	.5720	.5724	-	.5888	↑**	.5716	-	.5898	↑**
Both Power	.0990	.0927	-	.2729	↑**	.0990	-	.2708	↑**
Single Power	.4854	.4917	-	.7500	↑**	.4354	↓*	.7542	↑**
Co-Occur. Power	.1083	.0969	↓*	.0896	-	.1042	-	.0865	-
Generality	.6234	.6233	-	.6227	↓**	.6264	↑**	.6212	↓**
Macro Population	1754.3	1754.6	-	1740.5	↓**	1754.8	↑*	1738.7	↓**
Run Time (min)	3.70	2.57	↓**	2.53	↓*	2.64	↑*	2.59	↓*
200,000 Iterations (Ending Performance)									
Performance Statistics	UCS	ExSTraCS			AF		EK-AF		+QRF
	Core	Core	<i>p</i>	EK	<i>p</i>	AF	<i>p</i>	EK-AF	<i>p</i>
Training Accuracy	.8641	.8800	↑**	.8801	-	.8637	↓**	.8634	↓**
Test Accuracy	.5833	.5863	↑**	.5866	-	.5954	↑**	.5946	↑**
Both Power	.2583	.2563	-	.2625	-	.2906	↑**	.2948	↑**
Single Power	.6146	.6156	-	.6250	-	.5917	↓*	.5958	↓*
Co-Occur. Power	.1750	.1656	-	.1750	-	.1823	↑*	.1823	↑*
Rule Generality	.6946	.6945	-	.6945	-	.7501	↑**	.7518	↑**
Macro Population	1627.3	1627.6	-	1627.7	-	1444.8	↓**	1435.4	↓**
Run Time (min)	73.42	49.49	↓**	49.14	-	44.02	↓**	44.26	↓**

– No significant change

* $p < 0.05$ (Direction of change given by arrows)

** $p < 6.94 \times 10^{-4}$ (Cutoff assumes Bonferroni multiple test correction based on 72 comparisons)

EK and AF active in both algorithms (not shown), similar significant differences to those observed for ‘core’ comparisons were observed (i.e. ExSTraCS yielded faster run times and higher testing accuracy, but no difference in power).

Follow up analysis evaluated ExSTraCS with EK, and AF active on the continuous attribute simulation study. In short, we found that the adopted knowledge representation extends ExSTraCS to accommodate continuous attributes. Notably, the average run time for 200,000 learning iterations was significantly increased by about 30%, and performance (in terms of average testing accuracy and the three power metrics) was promising but significantly lower than performance in the discrete attribute datasets. This is likely because continuous attributes require ExSTraCS to learn not only ‘which’ attributes to specify, but appropriate interval ranges as well. Addressing these performance losses will be a target for ongoing research.

4 Conclusions

While ExSTraCS has been developed for biomedical, epidemiological, bioinformatics, and genetics problem domains in particular, we expect this new algorithm to be translatable to many related domains, and hopefully inspire new mechanisms and

improvements based on this core architecture. Through extensive simulation studies we have demonstrated the value of bringing successful mechanisms together in ExSTraCS in order to improve the key objectives of a successful data mining algorithm including speed, learning efficiency, flexibility, ease of use, scalability, and interpretability. In addition to improving continuous attribute performance, future work will address (1) expanding ExSTraCS to also accommodate continuous endpoints (e.g. quantitative traits), (2) further scalability (3) reassessment of fitness and deletion metrics to improve learning efficiency and (4) accessibility and usability through the development of ExSTraCS GUI software.

Acknowledgments. This work was supported by NIH grants AI59694, LM009012, LM010098, EY022300, LM011360, CA134286, and GM103534.

References

1. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: A complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications* (2009)
2. Thornton-Wells, T., Moore, J., Haines, J.: Genetics, statistics and human disease: Analytical retooling for complexity. *TRENDS in Genetics* 20(12), 640–647 (2004)
3. Wilson, S.: Classifier fitness based on accuracy. *Evo. Comp.* 3(2), 149–175 (1995)
4. Bernadó-Mansilla, E., Garrell-Guiu, J.: Accuracy-based learning classifier system: Models, analysis and applications to classification tasks. *Evo. Comp.* 11(3), 209–238 (2003)
5. Urbanowicz, R., Moore, J.: The application of michigan-style learning classifier systems to address genetic heterogeneity and epistasis in association studies. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 195–202. ACM (2010)
6. Urbanowicz, R., Moore, J.: The application of pittsburgh-style lcs to address genetic heterogeneity and epistasis in association studies. *Parallel Problem Solving from Nature-PPSN XI*, 404–413 (2011)
7. Urbanowicz, R., Granizo-Mackenzie, A., Moore, J.: Instance-linked attribute tracking and feedback for michigan-style supervised learning classifier systems. In: *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference*, pp. 927–934. ACM (2012)
8. Urbanowicz, R.J., Andrew, A.S., Karagas, M.R., Moore, J.H.: Role of genetic heterogeneity and epistasis in bladder cancer susceptibility and outcome: A LCS approach. *Journal of the American Medical Informatics Association* (2013)
9. Urbanowicz, R.J., Granizo-Mackenzie, D., Moore, J.H.: Using expert knowledge to guide covering and mutation in a michigan style LCS to detect epistasis and heterogeneity. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I. LNCS*, vol. 7491, pp. 266–275. Springer, Heidelberg (2012)
10. Tan, J., Moore, J., Urbanowicz, R.: Rapid rule compaction strategies for global knowledge discovery in a supervised learning classifier system. In: *Advances in Artificial Life, ECAL*, vol. 12, pp. 110–117 (2013)
11. Bacardit, J., Krasnogor, N.: A mixed discrete-continuous attribute list representation for large scale classification domains. In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pp. 1155–1162. ACM (2009)

12. Urbanowicz, R.J., Granizo-Mackenzie, A., Moore, J.H.: An analysis pipeline with statistical and visualization-guided knowledge discovery for michigan-style learning classifier systems. *IEEE Computational Intelligence Magazine* 7(4), 35–45 (2012)
13. DeJong, K.A., Spears, W.M.: Learning concept classification rules using genetic algorithms. Technical report, DTIC Document (1990)
14. Greene, C., Penrod, N., Kiralis, J., Moore, J.: Spatially uniform relief (surf) for computationally-efficient filtering of gene-gene interactions. *BioData Mining* 2(1), 1–9 (2009)
15. Kononenko, I.: Estimating attributes: analysis and extensions of relief. In: Bergadano, F., De Raedt, L. (eds.) *ECML 1994*. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994)
16. Greene, C.S., Himmelstein, D.S., Kiralis, J., Moore, J.H.: The informative extremes: Using both nearest and farthest individuals can improve relief algorithms in the domain of human genetics. In: Pizzuti, C., Ritchie, M.D., Giacobini, M. (eds.) *EvoBIO 2010*. LNCS, vol. 6023, pp. 182–193. Springer, Heidelberg (2010)
17. Granizo-Mackenzie, D., Moore, J.H.: Multiple threshold spatially uniform relief for the genetic analysis of complex human diseases. In: Vanneschi, L., Bush, W.S., Giacobini, M. (eds.) *EvoBIO 2013*. LNCS, vol. 7833, pp. 1–10. Springer, Heidelberg (2013)
18. Urbanowicz, R.J., Kiralis, J., Sinnott-Armstrong, N.A., Heberling, T., Fisher, J.M., Moore, J.H.: Gametes: A fast, direct algorithm for generating pure, strict, epistatic models with random architectures. *BioData Mining* 5(1), 16 (2012)

A Cooperative Evolutionary Approach to Learn Communities in Multilayer Networks

Alessia Amelio and Clara Pizzuti

National Research Council of Italy (CNR),
Institute for High Performance Computing and Networking (ICAR),
Via P. Bucci 41C, 87036 Rende (CS), Italy
{amelio,pizzuti}@icar.cnr.it

Abstract. In real-world complex systems objects are often involved in different kinds of connections, each expressing a different aspect of object activity. Multilayer networks, where each layer represents a type of relationship between a set of nodes, constitute a valid formalism to model such systems. In this paper a new approach based on Genetic Algorithms to detect community structure in multilayer networks is proposed. The method introduces an extension of the modularity concept and adopts a genetic representation of a multilayer network that allows cooperation and co-evolution of individuals, in order to find an optimal division of the network, shared among all the layers. Moreover, the algorithm relies on a label propagation mechanism and a local search strategy to refine the result quality. Experiments show the capability of the approach to obtain accurate community structures.

1 Introduction

In the last few years complex systems described as networks of nodes connected by different kinds of relationships are receiving a lot of attention. In fact, the approach adopted so far of aggregating the great variety of links connecting objects constituting a network, revealed its weaknesses because of loss of information caused by such a simplified view of a system. Real-life networked systems present multiple ties, each generally playing a different role and exhibiting a different type of strength among objects. Representing such systems by using a single type of interaction is a rough approximation of reality. A more apt modeling of such systems can be obtained by *multilayer* networks[5]. Kivela et al., [5] introduced the concept of multilayer network as the most general notion to model complex networks, including *multiplex* [7], *multirelational* [4], *multidimensional* [9,10,6]. A multilayer network can be viewed as a set of slice networks. Each slice, modeled as a graph, represents an aspect of the object activity, since an object may be involved in distinct activities with variable concern. In multilayer networks grouping actors by considering only one type of interaction may lead to inaccurate community structures because information that could come from all the interactions is discarded. The objective in a multilayer network is to uncover a shared community structure among objects such that a quality function be optimized for all the layers at the same time.

Proposals to find groups in multilayer networks can be found in [9,10,6,12,3]. In particular, Tang et al. [9,10] proposed a method, named *Principal Modularity Maximization (PMM)* that for each layer, first the *structural features*, corresponding to the top eigenvectors with positive eigenvalues, are extracted, then these features are combined to obtain latent communities.

In this paper a new method, named *MultiGA (Multilayer Genetic Algorithm)*, able to detect a shared community structure in a multilayer network, is proposed. *MultiGA* adopts a genetic representation of individuals that allows co-evolution and cooperation among all the network layers. An individual is composed by a number of elements equal to the number of layers. Each element represents a division of the corresponding layer in communities, and it is co-evolved with all the others by *learning* from them their community structure through the optimization of a fitness function that combines the modularity values of each layer. *MultiGA* relies also on a label propagation mechanism and a local search strategy. The former mechanism aggregates nodes having no connection in a layer to the community recurring most often among its neighbors in all the layers. The local search strategy, similarly to the Blondel et al. [1] method for single-layer networks, moves a node to one of its neighboring communities if an increase in modularity is obtained. Experiments on synthetic and real-world networks show that *MultiGA* is able to detect accurate community structures in multilayer networks. The paper is organized as follows. The next section introduces multilayer networks and formalizes the problem of community detection. Section 3 describes the proposed approach. In section 4 the results of the experiments are reported. Finally, Section 5 concludes the paper.

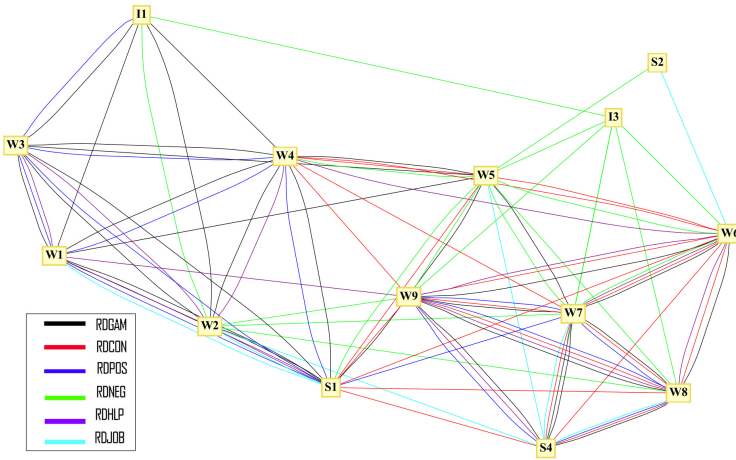


Fig. 1. The Roethlisberger & Dickson Bank Wiring Room of Western Electric multilayer network

2 Multilayer Networks

Let V be a set of n objects. A multilayer network is defined as a set $\mathcal{N} = \{\mathcal{N}_1, \dots, \mathcal{N}_d\}$ of *slice networks*. Each slice \mathcal{N}_s can be modeled as a graph $G_s = (V_s, E_s)$ where the

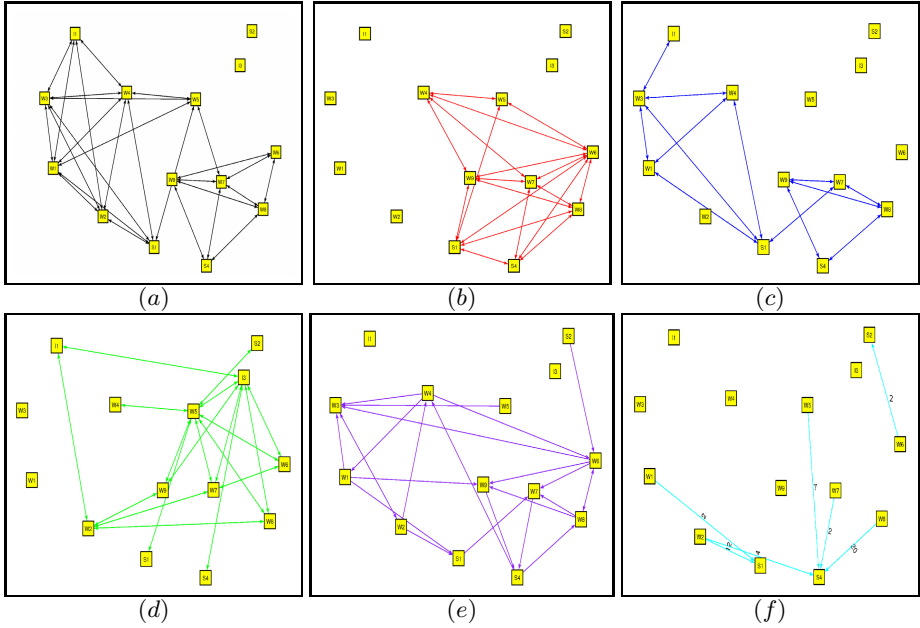


Fig. 2. (a) RDGAM relation, (b) RDCON relation, (c) RDPOS relation, (d) RDNEG relation, (e) RDHLP relation, (f) RDJOB relation

set of nodes $V_s \subseteq V$ is the subset of objects of V appearing in the slice \mathcal{N}_s , and E_s is the set of links that connect the objects of V_s in the s -th layer, i.e. an edge $(u, v) \in E_s$ if objects u and v interact in the s -th layer. \mathcal{N} can thus be represented as a set $\mathcal{G} = \{G_1, \dots, G_d\}$ of graphs, where each $G_s = (V_s, E_s)$, for $s = 1, \dots, d$, is the graph modeling network \mathcal{N}_s in the s -th layer. A layer thus represents one of the d slices of the network. Given an object $u \in V$, the neighbors of u at layer s are defined as $n_s(u) = \{v \in V_s \mid (u, v) \in E_s\}$, and the neighbors of u in \mathcal{G} as $n(u) = \cup_{s=1}^d n_s(u)$.

A clustering, or community structure, $\mathcal{CS}_s = \{C_{s_1}, \dots, C_{s_k}\}$ of a layer \mathcal{N}_s is a partitioning of G_s in groups of nodes that maximizes a quality function. Furthermore, for each couple of communities C_{s_i} and $C_{s_j} \in \mathcal{CS}_s$, $V_{s_i} \cap V_{s_j} = \emptyset$.

Our objective is to *uncover a shared community structure \mathcal{CS} among the objects of the multidimensional network \mathcal{N} such that a quality function is optimized in all the d dimensions*. An example of multilayer network is depicted in Figure 1. The example is taken from [2] and shows the relationships of 14 employees from a bank wiring room of Western Electric (Hawthorne Plant), downloaded from <http://moreno.ss.uci.edu/data.html>. The employees worked in a single room and include two inspectors (I1 and I3), three solderers (S1, S2 and S3), and nine wiremen or assemblers (W1 to W9). There are six different kinds of interactions among the employees: *RDGAM*, participation in horseplay; *RDCON*, participation in arguments about open windows; *RDPOS*, friendship; *RDNEG*, antagonistic behavior; *RDHLP*, helping others with work; and *RDJOB*, the number of times workers traded job assignments. The first

four types of connections are symmetric, while the last two aren't. Figures 2(a-f) show the six networks corresponding to each relation, where an unconnected node in a slice means that this node does not have any interaction of that type. For example, employee $S2$ has no ties with others regarding participation in horseplay ($RDGAM$) and arguments about open windows ($RDCON$), nor any friendship relation ($RDPOS$), i.e. $n_{RDGAM}(S2) = n_{RDCON}(S2) = n_{RDPOS}(S2) = \emptyset$, instead he has an antagonistic behavior with $W5$ ($RDNEG$), he helps $W6$ with work ($RDHLP$), and traded twice job assignment with him, i.e. $n_{RDNEG}(S2) = \{W5\}$, $n_{RDHLP}(S2) = \{W6\}$, but $n_{RDJOB}(S2) = \emptyset$ because there is an edge from $W6$ to $S2$ and not viceversa, thus $n(S2) = \{W5, W6\}$. The figure points out the intrinsic difficulty of grouping nodes in a proper way due to the incompleteness of information about the relations between two employees. In the next section an approach that combines the ties coming from all the layers is presented.

3 Method Description

In this section a detailed description of *MultiGA* is given, along with the genetic representation and operators adopted. Furthermore, the new concept of combined modularity is introduced and used as fitness function to optimize in order to search for a shared community structure in a multilayer network.

Genetic Representation and Operators. The genetic representation used by the approach is an extension of the locus-based adjacency representation. An individual $I = \{I_1, \dots, I_d\}$ of the population is composed by a set of d elements I_s , $1 \leq s \leq d$. Each element I_s consists of n genes g_1, \dots, g_n assuming integer values, corresponding to network nodes, in the range $\{1, \dots, n\}$. A value v assigned to the u -th gene means that there is a link between nodes u and v in the s -th graph G_s modeling the s -th network layer \mathcal{N}_s . If node u has no links in the s -th layer, i.e. $n_s(u) = \emptyset$, then it is assigned a zero value. Thus each element I_s of an individual I of the population gives a division of s -th layer \mathcal{N}_s of \mathcal{N} in a number c_s of communities.

I1	I3	W1	W2	W3	W4	W5	W6	W7	W8	W9	S1	S2	S4	
4	0	6	3	3	3	3	9	10	11	14	3	0	10	
0	0	0	0	0	7	8	12	8	8	6	11	0	8	
5	0	12	0	3	3	0	0	12	11	10	5	0	11	
4	8	0	1	0	7	2	7	2	2	2	7	7	2	
0	0	11	6	4	5	10	14	8	14	9	8	10		
0	0	12	12	0	0	14	13	14	14	0	0	0	0	
(a : Parent1)														
I1	I3	W1	W2	W3	W4	W5	W6	W7	W8	W9	S1	S2	S4	
6	0	5	3	3	3	3	9	10	11	9	7	0	9	
0	0	0	0	0	9	6	10	8	11	10	10	0	8	
5	0	12	0	3	3	0	0	10	14	10	3	0	10	
4	8	0	1	0	7	2	2	7	2	2	7	7	2	
0	0	12	6	4	3	5	10	14	8	14	9	8	10	
0	0	12	12	0	0	14	13	14	14	0	0	0	0	
(b : Parent2)														
I1	I3	W1	W2	W3	W4	W5	W6	W7	W8	W9	S1	S2	S4	
5	0	6	3	6	3	3	9	10	9	9	4	0	9	
0	0	0	0	0	8	8	9	6	8	8	8	0	8	
5	0	5	0	3	3	0	0	10	11	14	5	0	10	
4	9	0	1	0	7	2	2	2	2	2	4	7	2	
0	0	12	6	4	3	5	10	14	8	14	9	8	10	
0	0	12	12	0	0	14	13	14	14	0	0	0	0	
(c : before mutation)														
I1	I3	W1	W2	W3	W4	W5	W6	W7	W8	W9	S1	S2	S4	
0	0	1	1	0	1	1	0	1	1	0	0	0	0	
0	0	0	0	0	1	0	1	0	0	1	0	0	1	
1	0	1	0	0	0	0	0	1	1	0	0	0	0	
1	1	0	1	0	1	1	0	1	0	0	1	1	1	
0	0	1	1	0	0	1	1	0	0	0	1	1	0	
0	0	0	1	0	0	1	1	0	0	0	0	0	0	
(d : mask)														
I1	I3	W1	W2	W3	W4	W5	W6	W7	W8	W9	S1	S2	S4	
6	0	6	3	3	3	3	9	10	11	9	7	0	9	
0	0	0	0	0	7	6	12	8	11	6	10	0	8	
5	0	12	0	3	3	0	0	10	14	10	5	0	10	
4	8	0	1	0	7	2	7	2	2	2	7	7	2	
0	0	11	6	4	3	5	10	14	8	14	9	8	10	
0	0	12	12	0	0	14	13	14	14	0	0	0	0	
(e : child)														
I1	I3	W1	W2	W3	W4	W5	W6	W7	W8	W9	S1	S2	S4	
5	0	6	3	6	3	3	12	9	10	9	4	0	9	
0	0	0	0	0	8	12	9	6	8	8	8	0	8	
5	0	5	0	3	3	0	0	10	11	14	5	0	10	
4	9	0	1	0	7	13	2	2	2	2	4	7	2	
0	0	12	6	4	3	5	14	8	14	8	14	9	8	10
0	0	12	12	0	0	14	13	14	14	0	0	0	0	
(f : after mutation)														

Fig. 3. (a) First parent, (b) second parent, (c) individual before mutation, (d) binary mask, (e) individual after crossover, (f) individual after mutation

The initialization process, for every individual $I = \{I_1, \dots, I_d\}$, considers all the elements I_s , and assigns to a node u one of its neighbors v at random, where $v \in n_s(u)$, i.e. it is one of the neighboring nodes of u relative to the graph G_s corresponding to the s -th layer. If u has no neighbors in G_s , then the corresponding gene $g_u = 0$. Consider Figure 3(a) representing a generic individual $I = \{I_1, I_2, \dots, I_6\}$ of the Bank Wiring Room example consisting of six elements, i.e. the number of different relations. Employees have been numbered from 1 to 14, thus employee $I1$ is node 1, employee $I3$ is node 2, employee $W1$ is node 3, and so on. Every element I_s of I corresponds to a layer s and represents the graph G_s . For example I_2 , second row of Figure 3(a), represents the connections among employees with respect to relation $RDCON$. Nodes $I1, I3, W1, W2, W3, S2$ have no connections of type $RDCON$, thus the value of their neighbor node in I_2 is set to zero. $W4$ has neighbors $\{W5, W6, W7, W9\}$, as can be seen from Figure 2(b), thus I_2 at position 6 (corresponding to $W4$) has value 7, corresponding to $W5$, which is one of its neighbors.

The crossover operator is executed on each layer by applying uniform crossover. Given two parents $I = \{I_1, \dots, I_d\}$ and $J = \{J_1, \dots, J_d\}$, and a randomly generated binary vector, for each couple (I_s, J_s) uniform crossover selects the genes where the vector is a 1 from the first parent I_s , and the genes where the vector is a 0 from the second parent J_s , and combines the genes to form a child IJ_s . The crossover operator is showed in Figure 3. Consider, for example, the $RDGAM$ layer, and the corresponding parents I_1 , first row of Figure 3(a), and J_1 first row of Figure 3(b). The mask is that reported in the first row of Figure 3 (d). Thus the child IJ_1 generated by I_1 and J_1 , first row of Figure 3(e), is such that $IJ_1(1) = J_1(1) = 6$ and $IJ_1(2) = J_1(2) = 0$ because the mask is zero in the first two positions, while $IJ_1(3) = I_1(3) = 6$ because the mask value is 1 in the third position, and so on.

The mutation operator for every element I_s of $I = \{I_1, \dots, I_d\}$ assigns to each node u one of its neighbors $v \in n_s(u)$ at random. An example of mutation can be seen in Figure 3(c),(f). For example, consider relation $RDNEG$ (fourth row in Figure 3(c)). The neighbor of $W5$ is changed from 2 ($I3$) to 13 ($S2$), as can be seen in Figure 3(f).

Fitness Function. The choice of an appropriate fitness function is a key point to obtain a good solution for the problem to solve. As regards single-layer networks, the well known concept of *modularity* introduced by Girvan and Newman [8] is generally considered the one that at the best interprets the intuitive idea of dense group of nodes. The definition of modularity Q for single-layer networks is the following: $Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) \delta(C_i, C_j)$ where A is the adjacency matrix of the associated graph, m is the number of edges of the graph, k_i and k_j are the degrees of nodes i and j respectively. δ is the Kronecker function and yields 1 if i and j are in the same community (i.e. $C_i = C_j$), zero otherwise. Values approaching 1 indicate high quality clustering.

We propose to extend the concept of modularity to multilayer networks by combining the modularity values computed for each layer in such a way that the value for each layer is influenced by the values of all the other layers. Let s and r be two slices of a multilayer network, and $\mathcal{CS}_s, \mathcal{CS}_r$ be the clustering obtained on networks \mathcal{N}_s and \mathcal{N}_r respectively. Then the combined modularity Q_{sr} between slices s and r is defined as follows:

$$Q_{sr} = \frac{1}{2m_s + 2m_r} \sum_{ij} \left[(A_{ijs} - \frac{k_{is}k_{js}}{2m_s})\delta(C_{is}, C_{js}) + (A_{ijr} - \frac{k_{ir}k_{jr}}{2m_r})\delta(C_{is}, C_{js}) \right] \quad (1)$$

where A_s and A_r are the adjacency matrices of graphs G_s and G_r , respectively, k_{is} and k_{js} are the degrees of nodes i and j in G_s , while k_{ir} and k_{jr} are the degrees of nodes i and j in G_r , respectively. The Kronecker function δ yields 1 if i and j are in the same community C_s (i.e. $C_{is} = C_{js}$), zero otherwise. The meaning of Q_{sr} is that, while computing a community structure \mathcal{CS}_s on slice \mathcal{N}_s , this community structure is also checked on slice \mathcal{N}_r . Thus, if \mathcal{CS}_s does not determine a good grouping of nodes also in \mathcal{N}_r , it is penalized because the second term of Formula (1) is low.

Analogously, the *combined modularity* Q_{rs} between slices r and s is defined as

$$Q_{rs} = \frac{1}{2m_s + 2m_r} \sum_{ij} \left[(A_{ijr} - \frac{k_{ir}k_{jr}}{2m_r})\delta(C_{ir}, C_{jr}) + (A_{ijs} - \frac{k_{is}k_{js}}{2m_s})\delta(C_{ir}, C_{jr}) \right] \quad (2)$$

Finally, the *total combined modularity* Q_{ml} is computed on all the d slices by considering the sum of combined modularities Q_{sr} , for each couple of slices s and r :

$$Q_{ml} = \sum_{s, r \ s \neq r} (Q_{sr} + Q_{rs}) \quad (3)$$

In the next section a genetic algorithm that discovers shared community structure in multilayer networks by optimizing Q_{ml} is presented.

Algorithm Description. The evolutionary method we propose is based on the idea that, while detecting a community structure \mathcal{CS}_s on a layer s , it must be taken into account how much \mathcal{CS}_s is *similar* to the community structures \mathcal{CS}_r of the other layers, $r = 1, \dots, d, r \neq s$. The intuitive idea of similarity is that \mathcal{CS}_s contains groups of nodes that also appear in \mathcal{CS}_r , i. e. layers share communities. In order to pursue this objective, the algorithm *MultiGA*, thanks to the genetic representation that consists of individuals $I = \{I_1, \dots, I_d\}$ composed by a number d of elements, one for each layer, evolves individuals I by exchanging information among the layers. In fact, while searching for the division of a generic layer s in groups of nodes, by evolving the corresponding element $I_s \in I$, it *learns* from the other elements I_r $r = 1, \dots, d, r \neq s$ how much its clustering is shared with the other layers. This exchange of information is made possible by computing the total combined modularity value, that guides the search by exploiting the knowledge coming from all the slices. The *MultiGA* algorithm is described in Figure 4. It receives in input a multilayer network \mathcal{N} and gives a vector L containing a cluster labeling for each node of \mathcal{N} . *MultiGA* creates a random population of individuals $I = \{I_1, \dots, I_d\}$ (step 1) and evolves it for a fixed number of generations (step 2). For each individual in the population (step 3) the fitness function is calculated by using Formula (3) of the total combined modularity Q_{ml} . To this end, for each element I_s of I , the combined modularity of I_s with all the other layers is computed (steps 4-9). Then variation operators are applied and a new population created. At the end of the evolutionary process, a node label vector L_s is generated for each layer by assigning to each node the label of the community it belongs to, as determined by the clustering \mathcal{CS}_s (steps 13-17). If a node u in the layer s has not been assigned to any cluster because it has no links with nodes of that layer, then the *LabelAssignment* method (Figure 4(b)) considers its neighbors $n(u) = \cup_{s=1}^d n_s(u)$ in \mathcal{G} , and assigns to u the majority cluster

MultiGA Method:

Input: A multilayer network $\mathcal{N} = \{\mathcal{N}_1, \dots, \mathcal{N}_d\}$ of d dimensions, the set of graphs $\mathcal{G} = \{G_1, \dots, G_d\}$ modeling it

Output: A node cluster labeling L that partitions \mathcal{N} in the optimal shared community structure

- 1 **create** an initial population of random individuals $I = \{I_1, \dots, I_d\}$
- 2 **while** not maxGen
- 3 **for each** individual $I = \{I_1, \dots, I_d\}$
- 4 **decode** I and obtain partitionings $\mathcal{CS}_s = \{C_{s1}, \dots, C_{sk}\}$ for $s = 1, \dots, d$
- 5 $Q_{ml} = 0$
- 6 **for** $s = 1, \dots, d$
- 7 **compute** $Q_s = \sum Q_{sr} + Q_{rs}$, for $r = 1, \dots, d, r \neq s$
- 8 $Q_{ml} = Q_{ml} + Q_s$
- 9 **end for**
- 10 **end for**
- 11 **create** a new population by applying the variation operators
- 12 **end while**
- 13 **for** $s=1, \dots, d$
- 14 **initialize** the labeling vector L_s to null values
- 15 **for each** node v_j of \mathcal{G} appearing in $C_{si} \subset \mathcal{CS}_s$
- 16 **assign** cluster label si to v_j , i.e. $L_s(v_j) = si$
- 17 **end for**
- 18 **Perform** *LabelAssignment* on L_s
- 19 **end for**
- 20 **compute** the modularity value Q for each partitioning determined by L_s $s = 1, \dots, d$
- 22 **choose** as node label vector L the label vector L_s returning the maximum Q value;
- 21 **let** $\bar{\mathcal{G}} = \cup_{s=1}^d G_s$ be the graph obtained by joining all layers, where $\bar{A}_{ij} = 1$ if $\exists s$ such that $A_{ijs} = 1$
- 22 **Perform** *LocalSearch* on $\bar{\mathcal{G}}$ starting from solution L to improve modularity value Q

(a)

LabelAssignment Method:**Input:** the sequence of graphs $\mathcal{G} = \{G_1, \dots, G_d\}$ modeling \mathcal{N} and the node cluster labeling L_s of s -th layer**Output:** A node cluster labeling L_s where each node has been assigned a cluster label

- 1 **for each** node $u \in V$
- 2 **if** $(L_s(u) == 0)$
- 3 **let** $n(u) = \{v_{n_1}, \dots, v_{n_t}\}$ be the neigh. of u in \mathcal{G} and $L_s(v_{n_1}), \dots, L_s(v_{n_t})$ be the clust. label of v_{n_i} in \mathcal{CS}_s
- 4 $L_s(u) = \mathit{argmax} \{L_s(v_{n_1}), \dots, L_s(v_{n_t})\}$
- 5 **end if**
- 6 **end for**

(b)

Fig. 4. The pseudo-code of the *MultiGA* algorithm

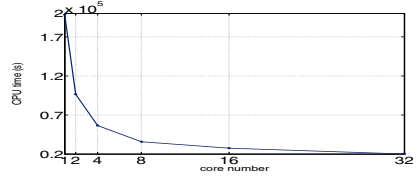
label in \mathcal{CS}_s of these neighbors, i.e. u is assigned the cluster label that occurs most often in \mathcal{CS}_s among its overall neighbors (steps 2-5 of *LabelAssignment* method). After that, for each layer s , the modularity value Q of Girvan and Newman [8] with respect to the partitioning determined by L_s is computed, and the labeling L_s giving the maximum Q value is chosen as final solution (steps 20-21). Finally, a post-processing local search, analogous to that proposed by Blondel et al. [1], is performed on the graph $\bar{\mathcal{G}} = \cup_{s=1}^d G_s$ obtained by the union of all the slices G_s where edges between two nodes are counted once, in order to improve modularity value. The local search, only once a time, moves a node to one of its neighboring communities if an increase in modularity, computed on the total graph $\bar{\mathcal{G}}$, is obtained. In the next section experiments on multilayer networks will show the feasibility of the approach in finding shared community structure.

Table 1. Average NMI values together with the standard deviation and best NMI values (in parenthesis) of *MultiGA* and a standard genetic algorithm using one slice at a time. Population size= $\{100, 500\}$, $\nu = 0.1, 0.3, 0.5$, $\mu = 0.5$. *nc* is the number of communities found.

Pop.	Strategy	$\nu = 0.1$	nc	$\nu = 0.3$	nc	$\nu = 0.5$	nc
100	A1	0.7629 \pm 0.071 (0.8751)	7 (34)	0.4620 \pm 0.124 (0.6684)	18 (145)	0.3696 \pm 0.116 (0.5560)	5 (22)
	A2	0.7277 \pm 0.110 (0.9116)	9 (20)	0.5345 \pm 0.101 (0.6267)	4 (8)	0.3939 \pm 0.158 (0.7063)	30 (262)
	A3	0.7997 \pm 0.067 (0.9303)	3 (5)	0.5461 \pm 0.151 (0.7092)	4 (6)	0.4451 \pm 0.177 (0.6681)	10 (43)
	A4	0.6421 \pm 0.167 (0.9097)	17 (53)	0.5179 \pm 0.118 (0.7080)	9 (30)	0.4801 \pm 0.157 (0.6667)	4 (11)
	<i>MultiGA</i>	0.9778 \pm 0.024 (1)	3 (3)	0.7793 \pm 0.139 (1)	2 (3)	0.7513 \pm 0.084 (0.8335)	2 (2)
500	A1	0.8568 \pm 0.088 (0.9498)	6 (35)	0.6663 \pm 0.200 (0.9498)	17 (145)	0.6076 \pm 0.105 (0.7529)	3 (10)
	A2	0.8307 \pm 0.087 (0.9498)	5 (11)	0.7617 \pm 0.087 (0.8846)	3 (3)	0.5883 \pm 0.141 (0.8134)	29 (262)
	A3	0.9048 \pm 0.081 (0.9707)	3 (3)	0.7627 \pm 0.072 (0.9383)	3 (5)	0.5833 \pm 0.174 (0.7371)	7 (43)
	A4	0.7237 \pm 0.174 (0.9414)	14 (44)	0.6653 \pm 0.088 (0.7795)	7 (36)	0.6562 \pm 0.064 (0.7304)	4 (9)
	<i>MultiGA</i>	0.9808 \pm 0.021 (1)	3 (3)	0.8376 \pm 0.139 (1)	2 (3)	0.7530 \pm 0.054 (0.7700)	2 (2)

	Strategy	Evolutionary			Spectral
		pop=100	pop=300	pop=500	
1-D	A1	0.7629 \pm 0.071	0.8484 \pm 0.082	0.8568 \pm 0.088	0.7237 \pm 0.1924
	A2	0.7277 \pm 0.110	0.7969 \pm 0.088	0.8307 \pm 0.087	0.6798 \pm 0.1888
	A3	0.7997 \pm 0.067	0.8338 \pm 0.111	0.9048 \pm 0.081	0.6672 \pm 0.1848
	A4	0.6421 \pm 0.167	0.7285 \pm 0.169	0.7237 \pm 0.174	0.6906 \pm 0.1976
	<i>MultiGA</i>	0.9778 \pm 0.024	0.9785 \pm 0.022	0.9808 \pm 0.021	-
4-D	<i>PMM</i>	-	-	-	0.9351 \pm 0.1059

(a)



(b)

Fig. 5. (a) Comparison of the NMI values between the evolutionary computation approaches and spectral approaches of Tang et al. [9]; (b) computation times required by *MultiGA* for increasing number of cores.

4 Experimental Results

This section provides a thorough experimentation for assessing the capability of *MultiGA* in detecting shared community structure in multilayer networks. The *MultiGA* algorithm has been written in MATLAB 7.14 R2012a, using the Genetic Algorithms and Direct Search Toolbox 2. A trial and error procedure has been adopted for fixing the parameter values. Thus the crossover rate has been fixed to 0.8, mutation rate to 0.2, elite reproduction 10% of the population size, number of generations is 150. We first present the results *MultiGA* obtained on randomly generated synthetic data sets. The networks have been generated as proposed by Tang et al. [9]. Each network is composed by 350 objects divided into three different clusters: the first one contains 50 objects, the second one 100 and the last one 200 objects. The objects are involved in $d = 4$ relations. A within-group probability μ connects the objects inside the same cluster. This probability value changes between groups at different slices. Any two nodes are connected to each other with probability ν , providing a controlled noise to the network. A clear network structure is obtained when the μ value is high and the ν value is low. 50 different synthetic networks have been randomly generated for different combinations of the parameters μ and ν and the average and standard deviation computed from the 50 runs. Since the ground truth partitioning of the nodes in communities is known a priori, in order to evaluate *MultiGA*, the Normalized Mutual Information (NMI) has been computed between the ground truth division in communities and the partitioning found

by *MultiGA*. Table 1 reports the average NMI values obtained by *MultiGA* together with standard deviation, and best NMI, in parenthesis, reached among all the runs. In order to show the superiority of *MultiGA* with respect to a naive approach that uses only one layer, the NMI values are compared with those returned by a standard genetic algorithm that optimizes Newman’s modularity by using only one layer at a time. For this experiment the within-group interaction parameter μ has been fixed to 0.5, while the noise parameter ν has been varied as 0.1, 0.3 and 0.5. Furthermore, we report the results for increasing values of population size, namely 100 and 500. In the table, nc denotes the average and the maximum, in parenthesis, number of communities found. The table clearly shows the very good results obtained by *MultiGA* that simultaneously evolves all the layers, with respect to running a naive method that finds a solution by using only one slice. This confirms the superiority of our technique with respect to single dimension based methods to discover community structure. It is worth to note that increasingly high NMI values are obtained by *MultiGA* at increasing values of population size, and often *MultiGA*, among the executions, is able to detect the ground truth division of the network. Moreover, when the noise is high, $\nu = 0.5$, the NMI values of *MultiGA* are never less than 0.75. Figure 5 (a) compares *MultiGA* with the method proposed by Tang et al. [9,10]. The table reports the results appearing in [9] for the spectral approach *PMM* that uses all the layers, and the spectral approach that uses a single layer at a time. The first observation is that the evolutionary approach always obtains higher NMI values with respect to the spectral approaches proposed by Tang et al. In particular, *MultiGA* reaches 0.97 with population size 100, and an even higher value of 0.98 when population size is 500, while the NMI value of *PMM* is 0.93. Analogously, the evolutionary strategy on single layers obtains better results than the spectral approach. It is worth pointing out that the spectral approach needs the number of communities as input parameter, while *MultiGA* automatically determines the number by optimizing the objective function. We also experiments *MultiGA* by fixing within-group interaction parameter $\mu = 0.8$. For lack of space we cannot report the overall results. However, for instance, in such a case, with population size 300, the NMI values are 0.75 for noise $\nu = 0.5$, 0.88 for $\nu = 0.3$ and 0.99 for $\nu = 0.1$. We performed experiments also on two real-life multilayer networks. The former is the Bank Wiring Room of Western Electric network of Figure 1. The grouping obtained consists of two communities, $\{W1, W2, W3, W4, I1, S1\}$ and $\{W5, W6, W7, W8, W9, I3, S4\}$ which seems rather plausible by observing Figure 1. The second one, is the famous multi-layer network consisting of marriage and business ties among 16 Florentine families in the 15th century [11], depicted in Figure 6. The figure shows the division we obtained in two groups (cyan and magenta respectively), and the isolated node *Pucci* in green, which has no connections in any of the two layers (as can be seen from Figure 6(b) and (c)). The division in two communities reflects the sharing of business and marriage relations. Moreover, it is worth to note that *Strozzi* and *Ridolfi* families have no marriage relationships, and they have been joined to the communities composed by $\{Lamberteschi, Guadagni, Castellani, Peruzzi, Bischieri\}$ and $\{Acciaiuli, Albizzi, Barbadori, Ginori, Medici, Pazzi, Salviati, Tornabuoni\}$ respectively, because they effectively have more business ties with the corresponding group. Thus *MultiGA* was able to properly capture the information coming from both types of relations. It is known

that genetic algorithms are naturally parallelizable. To this end, we used the Parallel Computing Toolbox of Matlab that allows multicore processing to deal with computationally intensive problems. We computed the times required by *MultiGA* on a computer cluster of 24 nodes, with 4 Gbyte of RAM and a 24-core Intel Xeon CPU at 2.6 GHz each, for a synthetic network of 5000 nodes with population size fixed to 300, when the number of cores used varies as 1, 2, 4, 8, 16, and 32. We obtained a linear speedup of the parallel implementation when 2 or 4 cores are employed (see Figure 5 (b)). In such a case, in fact, doubling the number of cores doubles the algorithm speed. When the number of cores increases to 8, 16, and 32 the speedup is almost linear, due to the times needed for communication. However, the time reduction is notably, going from 55 hours on one processor, to 5 hours when using 32 cores, showing that parallel implementation can give a valuable help in dealing with large networks.

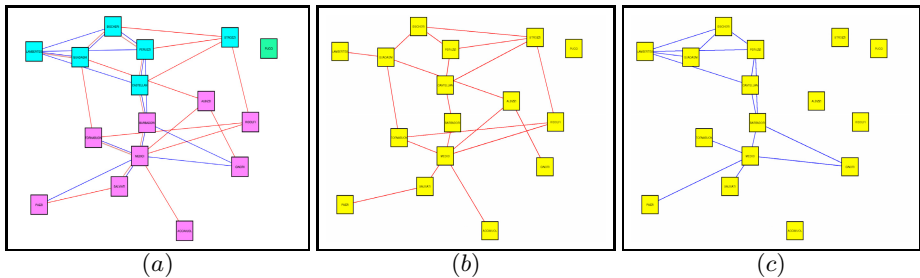


Fig. 6. (a) The Florentine families, (b) business relation (red), (c) marriage relation (blue)

5 Conclusions

The paper proposed a genetic algorithm to find shared community structure in multi-layer networks, based on the extension of genetic representation from single to multi-layer networks, and on the definition of total combined modularity concept. It employs two strategies, one to aggregate isolated nodes, and another to improve quality results by performing local search. Experiments on synthetic and real-life networks proved the capability of the approach to detect meaningful shared community structure.

References

1. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefevre, E.: Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, P10008 (2008)
2. Breiger, R.R., Boorman, S.A., Arabie, P.: An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. *Journal of Mathematical Psychology* 12, 328–383 (1975)
3. Comar, P.M., Tan, P.-N., Jain, A.K.: A framework for joint community detection across multiple related networks. *Neurocomputing* 76(1), 93–104 (2012)

4. Harrer, A., Schmidt, A.: Blockmodelling and role analysis in multi-relational networks. *Social Netw. Analys. Mining* 3(3), 701–719 (2013)
5. Kivela, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A.: Multilayer networks. [arXiv:1309.7233v3](https://arxiv.org/abs/1309.7233v3) (2014)
6. Li, X., Ng, M.K., Ye, Y.: Multicomm: Finding community structure in multi-dimensional networks. In: *IEEE Trans. on Knowl. and Data Eng.* (2013) (in press)
7. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.-P.: Community structure in time-dependent, multiscale, and multiplex networks. *Science* 328(5980), 876–878 (2010)
8. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Review E* 69, 026113 (2004)
9. Tang, L., Wang, X., Liu, H.: Uncovering groups via heterogeneous interaction analysis. In: *The Ninth IEEE Int. Conf. on Data Mining, ICDM 2009*, pp. 503–512 (2009)
10. Tang, L., Wang, X., Liu, H.: Community detection via heterogeneous interaction analysis. *Data Mining and Knowledge Discovery* 25(1), 1–33 (2012)
11. Wasserman, S., Faust, K.: *Social Network Analysis Methods and Applications*. Cambridge University Press (2009)
12. Zhang, Z., Li, Q., Zeng, D., Gao, H.: User community discovery from multi-relational networks. *Decision Support Systems* 54(2), 870–879 (2013)

Novelty Search in Competitive Coevolution

Jorge Gomes^{1,2}, Pedro Mariano², and Anders Lyhne Christensen^{1,3}

¹ Instituto de Telecomunicações, Lisbon, Portugal

² LabMAG – Faculdade de Ciências da Universidade de Lisboa, Portugal

³ Instituto Universitário de Lisboa (ISCTE-IUL), Lisbon, Portugal

jgomes@di.fc.ul.pt, plmariano@fc.ul.pt, anders.christensen@iscte.pt

Abstract. One of the main motivations for the use of competitive coevolution systems is their ability to capitalise on arms races between competing species to evolve increasingly sophisticated solutions. Such arms races can, however, be hard to sustain, and it has been shown that the competing species often converge prematurely to certain classes of behaviours. In this paper, we investigate if and how novelty search, an evolutionary technique driven by behavioural novelty, can overcome convergence in coevolution. We propose three methods for applying novelty search to coevolutionary systems with two species: (i) score both populations according to behavioural novelty; (ii) score one population according to novelty, and the other according to fitness; and (iii) score both populations with a combination of novelty and fitness. We evaluate the methods in a predator-prey pursuit task. Our results show that novelty-based approaches can evolve a significantly more diverse set of solutions, when compared to traditional fitness-based coevolution.

Keywords: Competitive coevolution, behavioural diversity, novelty search, convergence, evolutionary robotics.

1 Introduction

In a coevolutionary system, two or more populations simultaneously evolve, and evaluations are conducted with an individual from each population. In the case of competitive coevolution systems (CCESs), the populations represent competing species, where each succeed at the expense of the opposing species. CCESs are appealing for a number of reasons [18,14]: (i) they are suited to domains where a notion of absolute fitness might not exist; (ii) each population acts as a progressively more challenging opponent to the other population, and (iii) a CCES may be less prone to stagnation than non-coevolutionary methods, because of the ever-changing fitness landscape.

In practice, however, many of these advantages are frequently not observed [15]. The key to successful coevolutionary learning is a continuous arms race between the opposing species [7]. However, such an arms race is not easy to establish [6], and even when an arms race does occur, there is no guarantee that it will lead to good solutions. A number of techniques to help sustain an arms

race have been proposed [17], but most of them are focused on maintaining performance against opponents from earlier generations. While such techniques can contribute to a continuous and objective improvement of the solutions, they do not guarantee that a wide range of solutions are discovered, which is one of the most appealing arguments for the use of competitive coevolution [14]. The goal of a CCES is not necessarily to find the (near-)optimal solution to a problem [15]. The discovery of diverse solutions might be equally valuable, since there is no absolute measure of quality. Consider, for instance, the evolution of agents to play against human players: having a good diversity of playing strategies might be more valuable than having a single near-optimal player [19]. Moreover, it has been shown that promoting diversity in a population can potentially lead to improvements in the population's generalisation capacity [3].

Evolutionary techniques based on behavioural diversity, such as novelty search (NS) [11], have shown effective in overcoming behavioural convergence. These techniques drive evolution towards behavioural innovation, often resulting in a more effective evolutionary process that produces a greater diversity of solutions compared with traditional fitness-based evolution. Although most studies focus on non-coevolutionary domains [12,10], we recently showed that NS can be successfully used to overcome stable states in cooperative coevolution [8].

In this paper, we study the adaptation of behavioural diversity techniques to competitive coevolution. This combination of techniques is especially challenging because in a CCES, all populations are required to be fairly competitive at all times, which may conflict with the diversity objective [1]. We propose three different ways to apply novelty search to a CCES, and compare them with traditional fitness-based coevolution in a predator-prey pursuit task. We assess the proposed techniques along three dimensions: (i) the quality of the best solutions achieved; (ii) the exploration of the behaviour space; and (iii) the diversity of high-quality solutions evolved.

2 Related Work

2.1 Premature Convergence in Competitive Coevolution

Many reports of convergence of coevolving populations to undesirable regions of the solution space can be found in the literature (for examples, see [5,2,16]). One of the main causes for failure is that the coevolutionary process easily gets trapped in a *mediocre stable state* [7]: a cycle where a limited set of solutions is adopted by the populations over and over again. This cycle can give the impression of competition, without actually causing the evolutionary process to explore new solutions or to improve the quality of the individuals.

Another reason for the loss of diversity in the populations can be the lack of a fitness gradient [18]. It is possible that one population becomes so dominant that it turns into an *unhittable target* for its competitors. As such, the selection pressure disappears, and the populations stop improving. Ashlock et al. [1] tried to overcome the lack of a fitness gradient by rewarding individuals that generate a high variability of fitness scores in the opposing population. The approach

was, however, unsuccessful, since individuals that allow for a high variability of fitness typically do not pose a significant challenge.

A number of strategies for cultivating fruitful arms races have been proposed. Rosin & Belew [17] propose three techniques that aim to select a diverse and challenging set of competitors to test the individuals: (i) competitive fitness sharing, (ii) shared sampling, and (iii) the hall of fame. Other works have focused on the characteristics of the task setup that are favourable to the emergence and sustainability of arms races [13]. It has also been shown that shaping the environment or the fitness function throughout evolution can help avoid convergence to uninteresting and non-diverse solutions [5].

2.2 Novelty Search

Novelty search (NS) [11] is an evolutionary technique in which the population is driven towards behavioural novelty. NS has the potential to avoid premature convergence, and evolve a wide diversity of solutions in a single evolutionary run, as opposed to fitness-driven evolution that typically converges to a certain class of solutions [10]. The approach has been applied in the domain of evolutionary robotics and evolution of agent controllers with considerable success [12,10,8].

Novelty Search Algorithm. The distinctive aspect of NS is how the individuals of the population are scored. Instead of scoring individuals according to how well they perform a given task, which is typically measured by a fitness function, individuals are scored based on their behavioural novelty. Behavioural novelty is measured with a *novelty metric* that quantifies how different an individual is from other, previously evaluated individuals. The distance between two individuals is typically given by the distance between their *behaviour characterisation vectors*. These vectors are commonly composed of behavioural traits that the experimenter considers relevant for the task.

To measure how far an individual is from other individuals in behaviour space, the novelty metric relies on the average behaviour distance of that individual to the k nearest neighbours. Potential neighbours include the other individuals of the current generation and a random sample of individuals from previous generations (stored in an archive). Candidates from sparse regions of the behaviour space therefore receive higher novelty scores, which results in a constant evolutionary pressure towards behavioural innovation.

Combining Novelty and Fitness. As NS is guided by behavioural innovation alone, its performance can be greatly affected by the size and shape of the behaviour space [10]. Since NS is essentially an exploratory technique, most of the effort may be spent in behaviour regions that are irrelevant for the fulfilment of the task. Therefore, NS is often combined with fitness-based evolution to promote exploration of high-fitness behaviour regions.

In our experiments, we use a variant of *progressive minimal criteria novelty search* (PMCNS) [9] to combine novelty and fitness. PMCNS restricts exploration to behaviour regions associated with relatively high fitness scores. At each

generation g , the individuals' selection scores are assigned in the following way: if the fitness score of an individual is above the minimal criterion mc_g , its novelty score alone is used for selection, otherwise it receives a score of zero. The mc_g criterion corresponds to the P -th percentile value of the fitness scores of all individuals of generation g .

3 Approach

The application of behavioural diversity techniques to a CCES poses a number of challenges. In order to evolve effective solutions, coevolution requires an arms race between the coevolving species. This implies that all species are simultaneously improving to defeat one another: they learn to exploit flaws in the opponents' strategy. Behavioural diversity techniques are, however, essentially exploratory: evolution is driven towards behavioural novelty, not necessarily better solutions (i.e, solutions that are able to defeat the opponents). Prioritising diversity instead of competitiveness can compromise the effectiveness of coevolution, as shown in [1]. On the other hand, it has also been shown that when a population has converged to a strategy, it can be beneficial to reward alternate strategies, in order to foster the evolution of new behavioural traits [5].

We study the application of NS to a CCES with two competing species. The individuals' fitness score is measured in the traditional way: by competing against a representative sample of the individuals from the opposing species. In these competitions, we record the individual's behaviour as a task-specific behaviour characterisation vector. The novelty score of an individual is then calculated by computing the distance between its characterisation vector and the vectors of current and past individuals from the same species. The following methods are evaluated in this paper:

- Fit.** Selection based exclusively on the fitness score, in both populations. This method is used as a baseline.
- NS-Both.** Selection based exclusively on the novelty score in both populations.
- NS- p .** Selection based on the novelty score in population p (in our experiments, either *Pred* for predator, or *Prey*), and based on the fitness score in the other population.
- PMCNS.** Both populations are scored with PMCNS based on both novelty and fitness scores. The individuals are rewarded for displaying behavioural novelty while still meeting the minimal fitness criterion.

4 Experimental Setup

4.1 Predator-Prey Pursuit

The predator-prey pursuit task is a common testbed for CCESs [13,4]. In this task, two agent controllers are coevolved: one for a predator and one for a prey. The predator's objective is to capture (touch) the prey, and the prey's objective is to escape the predator. The agents operate in a closed square arena of $75 \times 75 \text{ cm}^2$.

A simulation trial ends if the predator captures the prey or if 100 seconds elapse. The initial conditions and setup of the task are depicted in Figure 1.

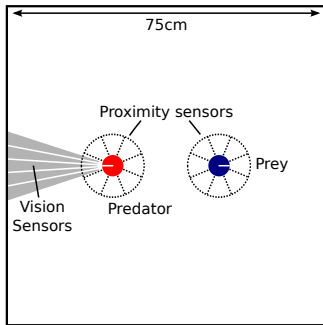


Fig. 1. Setup of the predator-prey task. The agents are initially placed in fixed positions, facing opposing directions.

The experimental setup is based on [14]. The prey and the predator move at a maximum speed of 5 cm/s. Both the prey and predator have eight proximity sensors, evenly distributed on their bodies, with a maximum range of 5 cm. The proximity sensors can detect both the walls and the other agent. The predator is additionally equipped with five vision sensors to detect the prey. The vision sensors have unlimited range and when combined provide a view angle of 40° . All sensors are binary: they return 1 if something is detected, and 0 otherwise. Each agent is controlled by a feed-forward neural network. Sensor readings are fed to the network, and two outputs control the speed of the two wheels. The networks of the predator and the prey have respectively 7 and 5 hidden neurons.

The fitness of the prey is given by the fraction of simulations it avoided capture. The fitness of the predator is the opposite: the fraction of simulations it was able to capture the prey. The behaviour characterisation used in the novelty-based setups is the same for both the predator and the prey. It comprises behavioural traits that are intuitively relevant in the context of the task. The characterisation is a vector of four elements: (i) the simulation length; (ii) the mean distance to the other agent throughout the simulation; (iii) the mean agent movement speed; and (iv) the mean distance between the agent and the closest wall. All elements are normalised to $[0, 1]$.

4.2 Coevolutionary Algorithm

The predator and prey controllers are coevolved in two separate populations. The weights of the neural controllers are directly encoded in the chromosomes. Each population is evolved with a simple evolutionary strategy with the following parameters: no crossover, a gene mutation rate of 5%, population size of 200, and tournament selection of size 5. At the end of each generation, the individual with the highest fitness score of each population is added to the respective *hall of fame* [17]. The individuals of each population are evaluated against a set of ten competitors randomly drawn from the opposing species' hall of fame.

NS is implemented as described in Section 2.2. Each population has its own novelty archive. Individuals are added to the archive with a probability of 3%. The archive can hold at most 1000 individuals: after reaching the limit, random individuals are removed to allow space for new ones. PMCNS was configured with

a percentile value P of 0.5, i.e., the median fitness of the population. Values of 0.25 and 0.75 were also tested, but yielded inferior performance.

5 Results

5.1 Quality of the Solutions

In all evolutionary runs, the individuals with the highest fitness score from both populations in every generation were recorded. To obtain an objective measure of the quality of these individuals, we performed a two-step master tournament [14], obtaining a *master fitness* and a *master behaviour characterisation*. First, the best individuals of each evolutionary run were identified by evaluating them against the individuals of the opposing species from the same run. Second, the individuals were evaluated against all the individuals identified in the first step, from all evolutionary treatments. The analysis of the individuals' master fitness can be seen in Figure 2 and Table 1 (Best fitness column).

These results show that all evolutionary treatments were able to achieve high-quality predator controllers. The differences between the treatments are relatively small. *Fit* is only significantly different from *NS-Both*, which is significantly inferior to all other treatments (Mann-Whitney U test, p -value < 0.01). The evolution of prey controllers, on the other hand, revealed larger differences between the evolutionary treatments. The treatment in which novelty had the biggest influence (*NS-Both*) was the lowest performing treatment with respect to the quality of the solutions (p -value < 0.01). Conversely, the treatments with highest fitness pressure yielded the highest scoring solutions, with none of the novelty-based treatments outperforming *Fit* (p -value < 0.01). This result confirms that in order to succeed, the populations need to have a strong selection pressure towards defeating the opposing population. In the predator-prey task, using NS to promote exploration of the behaviour space did not yield significant advantages regarding the achieved fitness scores.

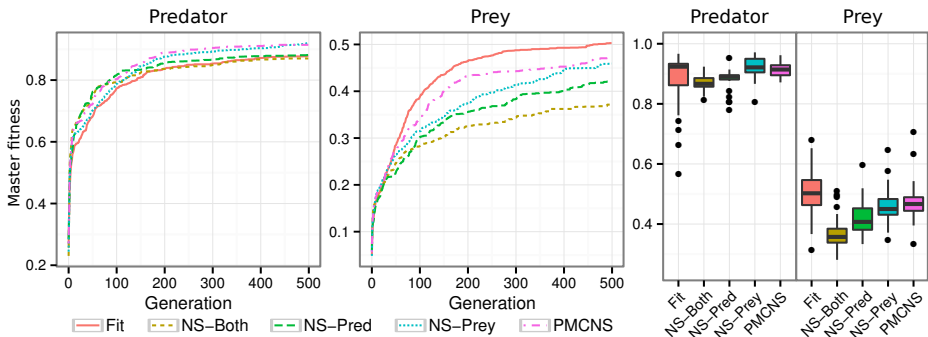


Fig. 2. Left and middle: best fitness found so far at each generation, for each population and each method. The fitness values correspond to the *master fitness*, and are averaged over 30 evolutionary runs. Right: boxplots of the highest fitness scores found in each evolutionary run.

Table 1. Best fitness found in each evolutionary run; Average coverage of the behaviour space in a whole evolutionary run; Average coverage of the high-fitness behaviour regions. Values are averages of 30 evolutionary runs for each method. Standard deviations are in parentheses.

	Best fitness		Global exploration		Elite exploration	
	Predator	Prey	Predator	Prey	Predator	Prey
<i>Fit</i>	0.88 (0.10)	0.50 (0.09)	0.42 (0.05)	0.37 (0.05)	0.49 (0.11)	0.46 (0.06)
<i>NS-Both</i>	0.87 (0.03)	0.37 (0.06)	0.64 (0.03)	0.60 (0.03)	0.62 (0.04)	0.60 (0.05)
<i>NS-Pred</i>	0.88 (0.04)	0.42 (0.06)	0.64 (0.02)	0.40 (0.04)	0.67 (0.06)	0.43 (0.08)
<i>NS-Prey</i>	0.92 (0.04)	0.46 (0.06)	0.43 (0.03)	0.59 (0.03)	0.50 (0.04)	0.70 (0.04)
<i>PMCNS</i>	0.91 (0.03)	0.48 (0.07)	0.59 (0.03)	0.53 (0.04)	0.70 (0.03)	0.59 (0.06)

5.2 Behaviour Space Exploration

One of the main advantages of novelty-based evolutionary techniques is the diversity of evolved solutions. To make a quantitative analysis of the behaviour space exploration, the space was first divided in regions of equal size: each dimension of the behaviour characterisation was discretised into 5 levels, resulting in a total of 625 regions. We then calculated how many times each region was visited throughout each evolutionary run (based on the *master characterisations*). This distribution was compared with the uniform distribution to obtain a measure of behaviour space coverage. The non-visited regions of the behaviour space (not reached by any method) were excluded, see Table 1 – Global exploration. To measure the exploration of the high-fitness behaviour regions, we followed the same procedure but excluded the regions where no reasonably good solutions were found (master fitness below 0.8 for the predator population, and 0.3 for the prey population), see Table 1 – Elite exploration.

Fit has the lowest degree of exploration, which suggests that it typically converges to a single class of solutions. *NS-Both*, on the other hand, has the highest degree of global exploration, but the same does not hold for elite exploration. The lack of elite exploration can explain the inferior performance of *NS-Both* regarding the achieved fitness scores: too much effort is spent on the exploration of low-fitness behaviour regions. The results show that *PMCNS* is able to avoid spending too much effort on exploration of low-fitness behaviours. The global exploration in *PMCNS* is still significantly higher than *Fit*, and it has one of the highest degrees of elite exploration.

Comparing *NS-p* to *Fit*, it is possible to observe that the fitness-driven population in *NS-p* has similar exploration and fitness scores (see in Table 1 – Prey: *NS-Pred*, *Fit*, and Predator: *NS-Prey*, *Fit*). The fitness-driven population is not significantly affected by the greater behavioural diversity in the opposing, novelty-driven population. On the other hand, if we compare the novelty-driven population with *NS-Both*, we can see that it is favourable to have one of the populations driven by fitness (see in Table 1 – Predator: *NS-Pred*, *NS-Both*, and Prey: *NS-Prey*, *NS-Both*). When a novelty-driven population competes against a fitness-driven population, it is able to achieve significantly higher elite exploration scores, which

also translates into higher fitness scores reached. By coevolving with a highly competitive, fitness-driven population, the novelty-driven population tends to move towards better solutions – even though their individuals are selected based on their novelty scores alone.

5.3 Diversity of Effective Solutions

The behaviour of the most effective preys is always very similar: they move at full speed in circular trajectories around the arena, using the proximity sensors to avoid the walls and to escape the predators when being chased. This behaviour typically fails when the predator quickly approaches the prey head on or from the sides, or when the prey is being chased and gets trapped between a wall and the predator. The set of successful predator strategies evolved was more diverse, especially in novelty-based treatments. To visualise the diversity of predator behaviours, we reduced the four dimensions of the characterisation using a Kohonen self-organising map. The Kohonen map was trained using a sample of the predator behaviours (*master characterisations*) found in all evolutionary runs. The solutions evolved in each run were then mapped individually. Figure 3 shows one typical evolutionary run of each treatment. *NS-Prey* is omitted since the exploration in the predator population is similar to *Fit*.

The results shown in the exploration maps are in accordance with the exploration scores (Table 1). The evolutionary runs of *Fit* typically converge to region B (23 out of 30 runs). In seven of those runs, solutions in region A were also discovered. Only one run of *Fit* explored regions C and D, although these regions are associated with solutions of similar quality to the solutions of regions A and B. *NS-Both* explored the behaviour space relatively uniformly, without any noticeable bias towards specific regions. *NS-Pred* is similar to *NS-Both*, but clearly spends more effort in high-fitness regions. The exploration pattern of the *PMCNS* treatment is in between that of *NS-Both* and *Fit*: a single evolutionary run typically converges to multiple classes of high-quality solutions.

Observing the predator solutions in action confirms that the novelty-based treatments can evolve an interesting and diverse set of behaviours. The following

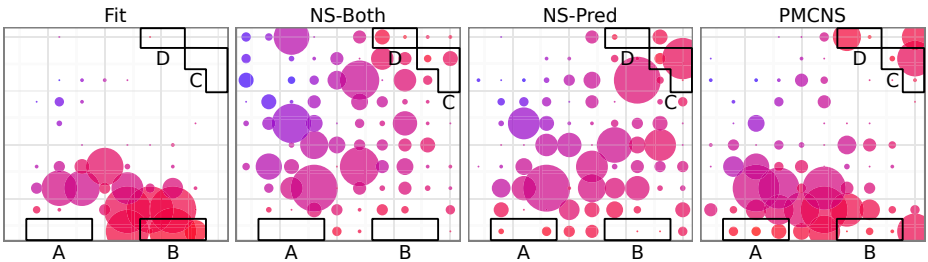


Fig. 3. Exploration of the predator behaviour space in typical evolutionary runs. The diameter of each circle is proportional to the number of individuals found belonging to that behaviour region. The highlighted regions A, B, C and D are associated with the highest fitness scores.

behaviour descriptions correspond to the highest scoring predator solutions found in each of the behaviour regions highlighted in Figure 3.

- (A) Moves backwards and in a curved trajectory until the prey is detected with the vision sensors. Afterwards, the predator stops and rotates to face the prey. When the prey gets close, the predator rushes towards it.
- (B) Moves forwards at full speed, avoiding the walls, until the vision sensors detect the prey. Afterwards, the predator chases the prey at full speed.
- (C) Moves slowly and directly towards a wall, and then stays there, facing the centre of the arena. When a prey passes nearby, the predator quickly rushes forward to catch the prey.
- (D) Does not initially move, only rotates to find the prey. When the prey is located, the predator keeps rotating to face the prey, and moves backwards very slowly. When the prey is near, the predator rushes towards it.

6 Conclusions

We proposed three methods to promote behavioural diversity in competitive coevolution: selection based on novelty score in both populations (*NS-Both*), novelty score only in one of the populations (*NS-p*), and a combination of novelty and fitness scores in both populations (*PMCNS*). These methods were compared with traditional fitness-based coevolution (*Fit*) in a predator-prey task.

With respect to the highest fitness scores achieved in the evolutionary runs, there was no significant advantage of novelty-based methods, when compared to fitness-based evolution. The novelty-based methods did, however, display significantly higher degrees of behaviour space exploration. The *NS-Both* method had the lowest performance among the novelty-based methods, since most of the exploration effort was spent in behaviour regions associated with relatively low fitness scores. However, when a novelty-driven population (*NS-p*) competed against a fitness-driven population, results showed that the novelty-driven population had a significantly stronger tendency to explore high-fitness behaviour regions. *PMCNS* was effective all-around: it could achieve fitness scores comparable to *Fit*, and could discover a wide range of high-quality solutions in a single evolutionary run, for both predator and prey controllers.

The novelty-based methods could consistently explore new behaviour regions and find classes of solutions that *Fit* rarely or never reached. In particular, a diverse set of interesting high-quality predator solutions was identified. Our experiments showed that novelty-based techniques can be used to avoid behavioural convergence and discover a broad diversity of solutions in competitive coevolution systems. The effectiveness of the novelty-based techniques depends on the balance between diversity and competitiveness, since the populations require challenging opponents in order to achieve solutions of high objective quality.

Acknowledgements. This research is supported by Fundação para a Ciência e Tecnologia (FCT) grants PEst-OE/EEI/LA0008/2013, PEst-OE/EEI/UI0434/2014, SFRH/BD/89095/2012 and EXPL/EEI-AUT/0329/2013.

References

1. Ashlock, D., Willson, S., Leahy, N.: Coevolution and tartarus. In: Congress on Evolutionary Computation, CEC, vol. 2, pp. 1618–1624. IEEE Press (2004)
2. Avery, P., Louis, S.: Coevolving team tactics for a real-time strategy game. In: Congress on Evolutionary Computation, CEC, pp. 1–8. IEEE Press (2010)
3. Chong, S.Y., Tino, P., Yao, X.: Relationship between generalization and diversity in coevolutionary learning. *IEEE Transactions on Computational Intelligence and AI in Games* 1(3), 214–232 (2009)
4. Cliff, D., Miller, G.F.: Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In: Morán, F., Merelo, J.J., Moreno, A., Chacon, P. (eds.) *ECAL 1995*. LNCS, vol. 929, pp. 200–218. Springer, Heidelberg (1995)
5. Dziuk, A., Miikkulainen, R.: Creating intelligent agents through shaping of coevolution. In: Congress on Evolutionary Computation, CEC, pp. 1077–1083. IEEE Press (2011)
6. Ebner, M., Watson, R.A., Alexander, J.: Coevolutionary dynamics of interacting species. In: Di Chio, C., et al. (eds.) *EvoApplicatons 2010, Part I*. LNCS, vol. 6024, pp. 1–10. Springer, Heidelberg (2010)
7. Ficici, S.G., Pollack, J.B.: Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In: *Artificial Life*, pp. 238–247. MIT Press (1998)
8. Gomes, J., Mariano, P., Christensen, A.L.: Avoiding convergence in cooperative coevolution with novelty search. In: *International Conference on Autonomous Agents and Multi-agent Systems, AAMAS*, pp. 1149–1156. IFAAMAS (2014)
9. Gomes, J., Urbano, P., Christensen, A.L.: Progressive minimal criteria novelty search. In: Pavón, J., Duque-Méndez, N.D., Fuentes-Fernández, R. (eds.) *IBERAMIA 2012*. LNCS, vol. 7637, pp. 281–290. Springer, Heidelberg (2012)
10. Gomes, J., Urbano, P., Christensen, A.: Evolution of swarm robotics systems with novelty search. *Swarm Intelligence* 7(2-3), 115–144 (2013)
11. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation* 19(2), 189–223 (2011)
12. Mouret, J.B., Doncieux, S.: Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary Computation* 20(1), 91–133 (2012)
13. Nolfi, S.: Co-evolving predator and prey robots. *Adaptive Behavior* 20(1), 10–15 (2012)
14. Nolfi, S., Floreano, D.: Coevolving predator and prey robots: Do arms races arise in artificial evolution? *Artificial Life* 4(4), 311–335 (1998)
15. Popovici, E., Bucci, A., Wiegand, R.P., De Jong, E.D.: Coevolutionary principles. In: *Handbook of Natural Computing*, pp. 987–1033. Springer (2012)
16. Reisinger, J., Bahçeci, E., Karpov, I., Miikkulainen, R.: Coevolving strategies for general game playing. In: *Computational Intelligence and Games*, pp. 320–327. IEEE Press (2007)
17. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. *Evolutionary Computation* 5(1), 1–29 (1997)
18. Watson, R.A., Pollack, J.B.: Coevolutionary dynamics in a minimal substrate. In: *Genetic and Evolutionary Computation Conference, GECCO*, pp. 702–709. Morgan Kaufmann (2001)
19. Yannakakis, G.N., Hallam, J.: Modeling and augmenting game entertainment through challenge and curiosity. *International Journal on Artificial Intelligence Tools* 16(6), 981–999 (2007)

An Immune-Inspired Algorithm for the Set Cover Problem

Ayush Joshi, Jonathan E. Rowe, and Christine Zarges

School of Computer Science, University of Birmingham,
Edgbaston, Birmingham, UK
{axj006, j.e.rowe, c.zarges}@cs.bham.ac.uk

Abstract. This paper introduces a novel parallel immune-inspired algorithm based on recent developments in the understanding of the germinal centre reaction in the immune system. Artificial immune systems are relatively new randomised search heuristics and work on parallelising them is still in its infancy. We compare our algorithm with a parallel implementation of a simple multi-objective evolutionary algorithm on benchmark instances of the set cover problem taken from the OR-library. We show that our algorithm finds feasible solutions faster than the evolutionary algorithm using less parameters and communication effort.

Keywords: Artificial immune systems, GSEMO, set cover.

1 Introduction

Artificial immune systems (AIS) are randomised search heuristics inspired from the immune system of vertebrates [3]. Unlike any other biological system, the immune system has several desirable properties combined together, which make it a great inspiration for the design of randomised search heuristics. Due to properties like diversity, robustness, and memory, algorithms inspired by the immune system have been applied to a large number of applications such as machine learning, security, robotics, and optimisation [3].

As more and more problems in the real world are getting increasingly complex the approaches to solve these are unable to scale and maintain robustness [7,10]. Natural processes on the other hand are robust and perform complicated tasks well, thus it is hoped that understanding and using more detailed ideas from these systems will help us design better systems. In this direction some work has been done by Greensmith [7] on the dendritic cell algorithm but this has been limited to intrusion detection and classification. Sim et al. [15] have proposed a hyper-heuristic called NELLI which learns from changing problem landscapes and has been shown to perform better than single human-designed heuristics.

In recent decades with the advancements in technology parallel architectures and multi-processor systems are becoming more and more common. As a consequence parallelisation of evolutionary algorithms (EA), in order to better utilise available resources and save time, is gaining importance and popularity [11]. EAs are inherently suitable for parallel implementations as operations such as fitness

calculations and mutations can be performed on separate processors. Parallel EAs based on island models, where each island is an independent population evolving on a separate processor, have features like inherent diversity. Some communication is required to guide the search process and this is achieved by exchange of solutions between islands.

We propose a novel artificial immune algorithm, *the germinal centre artificial immune system* (GC-AIS) which has been developed based on recent understanding of the germinal centre reaction in the immune system [16]. This new model has interesting properties like a dynamic population of islands and inherent parallelism. We compare it with a parallel version of the global simple evolutionary multi-objective optimiser (GSEMO) [12] on instances of the set cover problem taken from the OR-library [1]. It is shown that the GC-AIS is able to reach the feasible solution region faster than the homogeneous island model with less communication effort as well as less parameters to be set manually.

The outline of the paper is as follows: In Section 2 some preliminary information about the parallel GSEMO is provided along with the problem description. Section 3 gives the description of the GC-AIS model along with its pseudocode. In Section 4 the experimental set-up along with the obtained results are provided. The paper is concluded in Section 5 with a discussion on the observed results and conclusions thereafter.

2 Preliminaries

The set cover problem (SCP) can be defined as follows: Given a *universe set* U consisting of m items and a set S of n subsets of U whose union equals U , the goal is to find the smallest subset of S that covers the whole universe U . More formally:

Definition 1. *Let the set of m items $U := \{u_1, \dots, u_m\}$ denote the universe and let $S := \{s_1, \dots, s_n\}$ such that $s_i \subseteq U$ for $1 \leq i \leq n$ and $\bigcup_{i=1}^n s_i = U$. The unicost set cover problem can be defined as finding a selection $I \subseteq \{1, 2, \dots, n\}$ such that $\bigcup_{k \in I} s_k = U$ with minimum $|I|$.*

SCP is a NP-hard combinatorial optimisation problem with many practical applications, one of the most important being scheduling [2]. A survey of techniques used to solve the set cover problem can be found in [2]. The description of SCP above is a constrained single objective problem where the objective is to find the smallest subset of S which covers the universe and the constraint is that the subset covers the universe.

We convert this to a multi-objective version by using the constraint as a secondary objective [5]. Let vector $A = a_1 a_2 \dots a_n \in \{0, 1\}^n$ denote a solution where $a_i = 1$ if set s_i is in the solution and 0 otherwise. Let N equal the number of sub sets selected in A , and let C equal the number of elements left uncovered in U . The fitness function V for the SCP can now be written as a vector $V = \langle C, N \rangle$.

Multi-objective optimisation [4] is the task of finding optimal solutions to a problem which has several objectives, often competing with each other. A solution is said to dominate another if it is better in at least one objective and has at least the same fitness for the other objectives. In this case it is not always possible to order all individuals in a population since two potential solutions may each be good in a different objective and worse in the others. Therefore, there is not necessarily a unique optimal solution but a set of solutions where each member is not dominated by any other solution in the search space. This set is called the Pareto set and its image in the objective space is called the Pareto front.

The *global simple evolutionary multi-objective optimiser* (GSEMO) [6] is a generalisation of the (1 + 1) EA for multi-objective optimisation. It maintains a set of non-dominated solutions in every iteration. The parallel variant of GSEMO called the homogeneous island model GSEMO based on [12] can be described as a collection of μ islands which are fully connected to each other where each island runs an independent instance of the GSEMO algorithm. We refer to this model as parallel GSEMO (PGSEMO) throughout this paper. This is described in Algorithm 1.

Algorithm 1. Parallel GSEMO based on homogeneous island model [12]

Let P_i^t denote the population in each island i at generation t , μ denote number of islands, and p denote probability of communication.

Initialise $P^0 = \{P_1^0, \dots, P_\mu^0\}$ where $P_i^0 = \{0^n\}$ for $1 \leq i \leq \mu$. Let $t := 0$.

loop

for each island i in parallel **do**

 Select an individual x from P_i^t uniformly at random.

 Create offspring x' by mutating x with standard bit mutation, i. e., flip each bit with probability $1/n$.

if any individual in P_i^t dominates x' **then**

 Leave P_i^t unchanged.

else

 Remove all individuals dominated by x' from P_i^t and add x' to P_i^t .

end if

 With probability p send copy of population P_i^t to all $\mu - 1$ neighbours.

 Combine P_i^t with copies of populations received from neighbours.

 Remove all dominated solutions from P_i^t and let $P_i^{t+1} = P_i^t$.

end for

Let $t = t + 1$.

end loop

Communication effort can be described as the number of individuals which have been exchanged between the islands. We define the total *communication effort* as the total number of individuals which were transmitted in one run of the algorithm and the *parallel running time* as the number of generations it takes for the algorithm to reach an optimal solution [12].

3 The GC-AIS Algorithm

In the immune system [13] *germinal centres* (GC) are regions where the invading *antigen* (Ag) is presented to the *B cells* (a kind of immune cell) which create *antibodies* (Ab) that in turn bind to the pathogen in order to eradicate it. At the start of the invasion, the number of GCs grows and they try to find the best *Ab* for the pathogen by continuously mutating and selecting the B cells which can bind with the pathogen. Periodically GCs communicate by transmitting their *Abs* to other GCs. By proliferation, mutation and selection of immune cells this GC reaction is able to produce *Abs* which can eradicate the pathogen. Towards this stage the number of GCs starts declining.

The exact mechanism of selection in the GC is an active area of research and a new theory forms the basis of our algorithm [16]. According to this work the selection of B cells to be kept alive for proliferation, is maintained by the B cells themselves as they secrete *Ab* which bind with *Ag* and in turn directly compete with other B cells to bind with *Ag*. This is an inter-GC phenomenon as *Ab* from one GC can migrate to others and the competition increases which can lead to disappearance of GCs which can not cope up with the *Ab* from other GCs.

The motivation to apply the GC-AIS to the set cover problem comes from our belief that in an abstract way the immune system tries to solve the set cover problem. Every time the body is invaded by a pathogen, the immune system must produce *Abs* which are able to bind with the antigen (a partial cover) and must improve this *Ab* by optimisation so that the binding is strong enough to eradicate it (full cover). So if we visualise a possible pathogen binding site as an instance of the universe set, and the binding regions of the B cells as possible solutions, then the immune system tries to solve the problem of finding the best match to the pathogen, by randomised variations in the solutions.

The GC-AIS (see Algorithm 2) starts with one GC which contains one B cell, representing a problem solution. Offspring are created by standard bit mutation of B cells in GCs. In the current version of our algorithm we restrict ourselves to GCs that contain only a single B cell. In every generation there is a migration of *Ab* between GCs, performed by transmitting only the fitness value of the offspring from one GC to another. After migration, dominated solutions are deleted which can lead to the eradication of a GC. The surviving offspring form new GCs. This leads to a model where the number of GCs is dynamic in nature.

The GC-AIS always maintains a set of non-dominated solutions in every generation. A parameter for the number of GCs is not required as the number is dynamic and evolves as the algorithm runs. A preliminary design of the GC-AIS can be found in [9].

4 Experimental Results

In this section we present the results obtained on running the GC-AIS and the PGSEMO on some benchmark test instances of the SCP. The instances are

Algorithm 2. The GC-AIS

Let G^t denote the population of GCs at generation t and g_i^t the i -th GC in G^t .
 Create GC pool $G^0 = \{g_1^0\}$ and initialise g_1^0 . Let $t := 0$.
loop
 for each GC g_i^t in pool G^t in parallel **do**
 Create offspring y_i of individual g_i^t by standard bit mutation.
 end for
 Add all y_i to G^t , remove all dominated solutions from G^t and let $G^{t+1} = G^t$.
 Let $t = t + 1$.
end loop

selected from the SCP test bed of the OR-library [1] where the instances are grouped into classes based on the size of the problem. One instance each was selected randomly from the 12 problem classes named 4, 5, 6, A, B, C, D, E, RE, RF, RG and RH.

The PGSEMO requires the number of islands and the probability of communication to be set manually while GC-AIS does not require these parameters. As in [12], both algorithms initialise individuals in 0^n . This was done as most problem specific algorithms use this method. For the GC-AIS we observed that starting from a random string gives poor results. The stopping criteria can be based on a fixed budget of generations or letting the algorithms run until a certain desired fitness is achieved.

The probability of communication in the PGSEMO was initially based on the equation $p = \mu/mn$, where μ is the number of islands, p is the probability of communication and m and n describe the problem size. This value gives the best performance guarantee for a complete topology [12]. To estimate the number of islands μ for the PGSEMO, GC-AIS was run for 10,000 generations and it was observed that sufficiently good solutions were achieved. The average of the maximum number of islands in 30 runs was used as the number of islands for PGSEMO. This is done so that a fair comparison can be made in terms of computation resources available to both algorithms. Due to space restrictions it is not possible to include plots for every instance in this paper, key representative plots are provided.

The first set of experiments was performed to analyse the solution quality both algorithms can achieve using a fixed budget of fitness evaluations. A quota of 7500 generations was set as a stopping criteria and average fitness achieved per generation was plotted. This can be seen in Figures 1 and 2. It was observed experimentally that using $p = \mu/mn$ to set p resulted in sub-par performance and experiments were tried with higher rates of communication, $p = 1/n$, $p = 1/m$ and $p = 1/\mu$. These are shown for problem *scp41* in Figure 1. We observed that having the probability of communication $p = 1/\mu$, so that on average every generation one island communicates gives best results.

Figures 1 and 2 show the fitness achieved per generation in both the GC-AIS and the PGSEMO. The dotted lines depict the average number of sets used and the solid lines depict the average number of uncovered elements per generation.

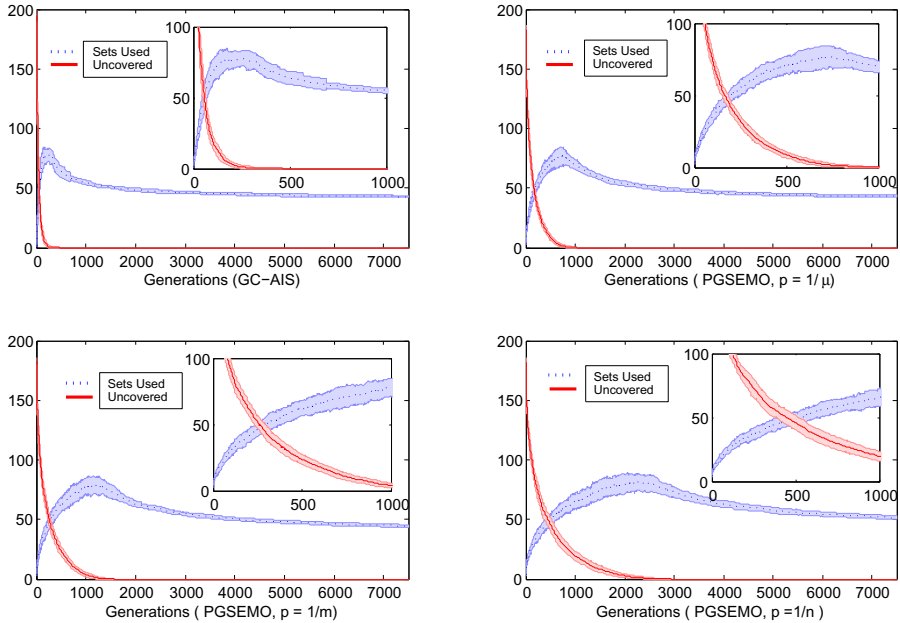


Fig. 1. Fitness plots for GC-AIS and PGSEMO for problem *scp41* with standard deviation as shaded error bars, averages performed over 30 independent runs

The standard deviation per generation can be seen as the shaded error-bars. To better see the difference between the curves in the early phase, the plots have been zoomed in, which can be seen as the smaller sub-plots inside these figures.

For our next set of experiments we are interested in finding the generations required to reach a fixed fitness value. From our previous experiment we use the best fitness value which has been achieved in every run. Average generations to reach this value were computed and the Wilcoxon rank-sum test was performed. We additionally compare the results achieved by PGSEMO and GC-AIS with the best known results and the results of a simple Greedy heuristic [8,14]. All results are shown in Table 1. In 8 out of the 12 rows of the table it can be seen that there is a significant difference (p -value smaller than 0.05) between the performance of the two algorithms, visible from the Wilcoxon rank-sum test results, these entries have been written in bold face. In 7 out of these 8 cases GC-AIS performed faster than PGSEMO.

To estimate the communication effort of the two algorithms, we count the number of individuals which are exchanged between islands per generation. The plots for the accumulated number of communications until generation t are shown in Figure 3 for the problem instances *scp41* and *scpbnr4*.

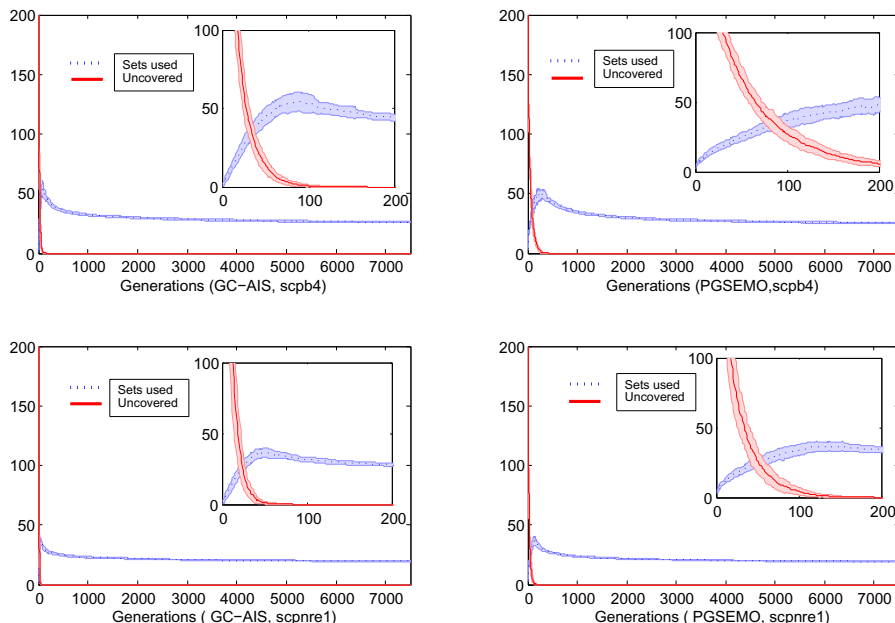


Fig. 2. Fitness plots for GC-AIS and PGSEMO for problem *scpb4* and *scpnre1* with standard deviation as shaded error bars, averages performed over 30 independent runs

Table 1. Generations required to reach a sufficiently good fitness. ‘AIS’ represents generations required for GC-AIS, ‘PGSEMO’ represents generations required for PGSEMO, ‘fitness’ is the target fitness value used as stopping criterion. The column ‘WRStest’ shows the p -values of the Wilcoxon rank-sum test, ‘Gr’ contains the fitness achieved by the Greedy heuristic, ‘known’ contains the best known value and ‘achieved’ contains best value achieved by GC-AIS in the first set of experiments. Generations are averaged over 30 runs.

Problem	$m \times n$	Fitness μ	AIS	PGSEMO	WRStest	Gr	Known	Achieved
scp41	200×1000	(0,45)	65 3654.5	4349.3	0.0013	41	38	41
scp52	200×2000	(0,43)	65 3412.4	4440.4	4.1127e-07	38	34	39
scp63	200×1000	(0,25)	45 2260.8	2037.3	0.3112	21	21	22
scpa5	300×3000	(0,50)	75 3518.8	4702.8	9.2603e-09	43	38	44
scpb4	300×3000	(0,28)	50 2941.2	2682.3	0.0575	24	22	25
scpc3	400×4000	(0,58)	90 3418.	4765.4	3.6897e-11	47	43	51
scpd2	400×4000	(0,31)	50 3507.8	3450.4	0.9646	26	25	28
scpe1	50×500	(0,5)	12 961.4	2051.1	0.0058	5	5	5
scpnre1	500×5000	(0,22)	30 1409	1506	0.1370	18	17	19
scpnrf2	500×5000	(0,12)	20 1867.5	1490.9	0.0302	11	10	11
scpnrg3	1000×10000	(0,87)	120 3168.4	6519.8	3.0199e-11	-	62	77
scpnrh4	1000×10000	(0,44)	65 3179	4069	1.3848e-06	-	34	40

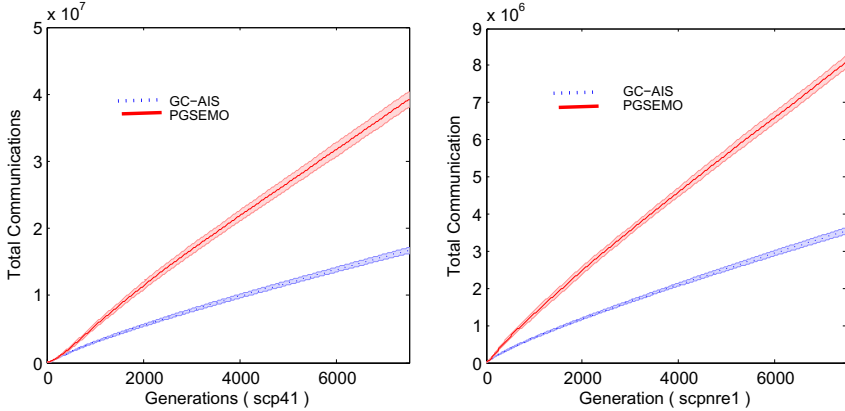


Fig. 3. Total communication cost until generation t , for problems *scp41* and *scpnr1* averaged for 30 runs, for GC-AIS and PGSEMO

5 Discussion and Conclusion

The GC-AIS is able to reach the region of feasible solutions, i.e., solutions where the first objective value is 0, faster than the PGSEMO. This can be seen in Figures 1 and 2: in the first 500-1000 generations the solid curve, which depicts the uncovered elements, can be seen to reach 0 faster for GC-AIS than for PGSEMO. It was observed that during the generations towards the end, mutations of individuals in the population very rarely replace any parent. We think that at this stage that all islands in the PGSEMO converged to a similar Pareto set due to communication over the course of the run, while there is in fact just one Pareto set for the GC-AIS. Therefore for the PGSEMO, having many parallel islands each with a similar population increases the chance of finding an improvement, in comparison to a single GC population in the GC-AIS. This advantage of having many islands comes at a cost, which is the communication effort. As communication increases the benefits of parallelism begin to fade, as it becomes a substantial time constraint for the overall performance. The GC-AIS requires far less communications over all than PGSEMO which can be seen in Figure 3. GC-AIS also uses less communication information than the PGSEMO, as only fitness values are sent to other GCs in GC-AIS while the whole population is communicated in PGSEMO.

Parameter setting is a big factor when running an algorithm on a new problem. The GC-AIS has a clear advantage over PGSEMO in terms of parameters to be set: the PGSEMO needs two parameters p and μ to be set manually while GC-AIS does not require any of these. As can be seen in Figure 1, setting the right values for p is crucial to obtain the desired performance. The values we found optimal are different from the ones, which give the best proven performance guarantees, as suggested in [12].

We proposed a novel immune-inspired algorithm called GC-AIS and compared it with a simple multi-objective evolutionary algorithm. With new ideas taken from

the immune system and an interesting motivation to use the set cover problem as a test, we show that the GC-AIS performs faster, uses less communication and has the advantage of not requiring as much human intervention to set it up. In the future we will investigate how the GC-AIS performs on other problem classes and compare it with state-of-the-art techniques for these problems.

References

1. Beasley, J.E.: OR-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society* 41(11), 1069–1072 (1990), <https://files.nyu.edu/jeb21/public/jeb/info.html>
2. Caprara, A., Toth, P., Fischetti, M.: Algorithms for the set covering problem. *Annals of Operations Research* 98(1-4), 353–371 (2000)
3. De Castro, L.N., Timmis, J.: *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer (2002)
4. Deb, K.: *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley-Blackwell (2001)
5. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation* 18(4), 617–633 (2010)
6. Giel, O., Wegener, I.: Evolutionary algorithms and the maximum matching problem. In: Alt, H., Habib, M. (eds.) *STACS 2003*. LNCS, vol. 2607, pp. 415–426. Springer, Heidelberg (2003)
7. Greensmith, J.: *The Dendritic Cell Algorithm*. PhD thesis, University of Nottingham (2007), <http://www.cs.nott.ac.uk/~jqg/thesis.pdf>
8. Grossman, T., Wool, A.: Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research* 101(1), 81–92 (1997)
9. Joshi, A.: Design of a parallel immune algorithm based on the germinal center reaction. In: *Proc of GECCO Companion*, pp. 1671–1674. ACM (2013)
10. Kim, J., Bentley, P.J.: Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. In: *Proc. of CEC*, vol. 2, pp. 1244–1252. IEEE Press (2002)
11. Luque, G., Alba, E.: *Parallel Genetic Algorithms: Theory and Real World Applications*. Springer (2011)
12. Mambrini, A., Sudholt, D., Yao, X.: Homogeneous and heterogeneous island models for the set cover problem. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I*. LNCS, vol. 7491, pp. 11–20. Springer, Heidelberg (2012)
13. Murphy, K.: *Janeway’s Immunobiology*. Garland Science (2011)
14. Musliu, N.: Local search algorithm for unicost set covering problem. In: Ali, M., Dapoigny, R. (eds.) *IEA/AIE 2006*. LNCS (LNAI), vol. 4031, pp. 302–311. Springer, Heidelberg (2006)
15. Sim, K., Hart, E., Paechter, B.: A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation* (to appear, 2014), http://dx.doi.org/10.1162/EVCO_a_00121
16. Zhang, Y., Meyer-Hermann, M., George, L.A., Figge, M.T., Khan, M., Goodall, M., Young, S.P., Reynolds, A., Falciani, F., Waisman, A., Notley, C.A., Ehrenstein, M.R., Kosco-Vilbois, M., Toellner, K.-M.: Germinal center B cells govern their own fate via antibody feedback. *The Journal of Experimental Medicine* 210(3), 457–464 (2013)

Natural Gradient Approach for Linearly Constrained Continuous Optimization

Youhei Akimoto¹ and Shinichi Shirakawa²

¹ Faculty of Engineering, Shinshu University, Nagano, Nagano, Japan
y_akimoto@shinshu-u.ac.jp

² College of Science and Engineering, Aoyama Gakuin University,
Sagamihara, Kanagawa, Japan
shirakawa@it.aoyama.ac.jp

Abstract. When a feasible set of an optimization problem is a proper subset of a multidimensional real space and the optimum of the problem is located on or near the boundary of the feasible set, most evolutionary algorithms require a constraint handling machinery to generate better candidate solutions in the feasible set. However, some standard constraint handling such as a resampling strategy affects the distribution of the candidate solutions; the distribution is truncated into the feasible set. Then, the statistical meaning of the update of the distribution parameters will change. To construct the parameter update rule for the covariance matrix adaptation evolution strategy from the same principle as unconstrained cases, namely the natural gradient principle, we derive the natural gradient of the log-likelihood of the Gaussian distribution truncated into a linearly constrained feasible set. We analyze the novel parameter update on a minimization of a spherical function with a linear constraint.

1 Introduction

The covariance matrix adaptation evolution strategy (CMA-ES) is a state-of-the-art randomized search heuristics in continuous domain [8–10, 12]. The CMA-ES maintains the Gaussian distribution, from which candidate solutions are drawn. It repeats the following: sample λ points from the Gaussian distribution, evaluate the fitness for each sample, update the parameters including the mean vector and the covariance matrix of the distribution in order to make the distribution likely to generate better solutions. Recently, it has been shown [2, 7] that the parameter update in the CMA-ES is partially interpreted as a natural gradient ascent on the parameter space of the Gaussian distribution, where the natural gradient is computed for the function defined below in (1). This idea is further generalized to the generic framework for arbitrary optimization, namely information-geometric optimization (IGO) [16].

Since the CMA-ES has been originally proposed for unconstrained continuous optimization, it often requires a treatment when solving a constrained problem. A number of constraint handling strategies have been proposed for evolution strategies and for more generic evolutionary algorithms [15]; e.g., adding an adaptive penalty to the fitness according to the constraint violation [11], repairing an infeasible point into the feasible region by a projection onto the boundary [4] or by a gradient based repair operator [13]. In this paper we consider the *resampling* strategy; an infeasible point is

discarded and resampled until it drops into the feasible region. It can be applied even when the constraint functions are black-box.

It has been shown that under a linear constraint, the success probability, i.e. the probability of generating a better point, depends on the angle of the gradients of the constraint function and the objective function and the dependency of the success probability on the angle is different for the resampling [5] and a repair operator [4]. This obviously affects a success probability based parameter update such as step-size adaptation based on the 1/5 success rule [17]. Moreover, since the distribution of the generated point in the feasible region is truncated (in the case of resampling) or biased on the boundary (in the case of repairing), the update rules that are designed from a statistical viewpoint are affected. For example, when the original Gaussian distribution is parameterised by the mean vector m and the covariance matrix C , these parameters no more represent the mean vector and the covariance matrix of the truncated Gaussian distribution. Then the maximum likelihood estimators for m and C for the truncated distribution differ from the ones for the original distribution. Therefore, a treatment in parameter update is required, an example of which is proposed by [6] where the covariance matrix is actively reduced in the direction of the gradient of the constraint.

In this paper we study the effect of the constraint from a viewpoint of the natural gradient. When resampling method is employed, the distribution of the generated feasible points is a truncated probability distribution whose domain is limited to the feasible set. In this situation the natural gradient differs from the one computed for the non-truncated (original) probability distribution. Now a question arises as to if we can gain a better performance by computing the natural gradient on the manifold of the truncated Gaussian distributions limited to the feasible set.

To address the question we derive the natural gradient under a linearly constrained feasible domain and compare it with the original natural gradient theoretically and numerically. In Section 2 the IGO framework and the rank- μ update CMA-ES as an instantiation of the IGO are revisited. In Section 3 we derive the natural gradient under a linearly constrained feasible domain. In Section 4 we analyze the infinite-population model using the derived natural gradient on a linearly constrained spherical problem and perform simulations to compare the behavior of the derived algorithm with the original algorithm on a linearly constrained spherical problem. Finally in Section 5 we summarize this work and discuss required future works.

Notation. The inner product of $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$ is denoted by $\langle x, y \rangle$ and the norm of x by $\|x\| = \langle x, x \rangle^{1/2}$. For any matrix A , $[A]_{i,j}$ represents the (i, j) th element, $[A]_{i,:}$ (or $[A]_{:,j}$) the i th row (or the j th column, respectively.) For any symmetric matrix A of dimension d , let $\text{vech}(A)$ denote the lower-left half vectorization of A such that $\text{vech}(A)$ is the $d(d + 1)/2$ dimensional column vector whose i th element is $[A]_{m_i, n_i}$ where $i = m_i + (d - n_i/2)(n_i - 1)$ for $1 \leq n_i \leq m_i \leq d$. We refer to [14] for the detail.

2 IGO Framework and the Rank- μ Update CMA

Formulation. We consider a constrained continuous minimization $\text{argmin}_{x \in X} f(x)$, where $X \subset \mathbb{R}^d$ is the feasible set and f is the objective function defined over X . In the

following sections we assume that the feasible set is restricted by a linear function, namely $X = \{x \in \mathbb{R}^d \mid \langle x, v \rangle \geq \alpha\}$ for some unit vector $v \in \mathbb{R}^d \setminus \{0\}$ and some $\alpha \in \mathbb{R}$.

Given a family \mathcal{P} of probability distributions P_θ on X parameterized by a real vector $\theta \in \Theta$, the IGO framework formulates the joint problem on the parameter space Θ at each iteration t as follows

$$\theta^{t+1} = \operatorname{argmax}_{\theta \in \Theta} J_{\theta^t}(\theta), \quad \text{where } J_{\theta^t}(\theta) := \int_X W_{\theta^t}^f(x) P_\theta(dx) . \tag{1}$$

Here θ^t is the value of the parameter at iteration t , $W_{\theta^t}^f$ defines the preference that is monotonic to f . The preference is defined based on the probability of sampling a better point; namely,

$$W_{\theta^t}^f(x) = w(P_{\theta^t}[y \in X \mid f(y) \leq f(x)]) , \quad \text{where } w : [0, 1] \rightarrow \mathbb{R} . \tag{2}$$

Another weight scheme is introduced in [1],

$$W_{\theta^t}^f(x) = -(\mu_{\text{Leb}}[y \in X : f(y) \leq f(x)])^{2/d} , \tag{3}$$

where μ_{Leb} denotes the Lebesgue measure on \mathbb{R}^d . This is theoretically attractive; on an unconstrained monotonic convex quadratic composite function $g(x^T A x)$ with g strictly increasing, this weight value is $-c x^T A x$, where c is a constant independent of g , m , and C , and it enables us to derive the exact $J_{\theta^t}(\theta)$.

Natural Gradient. The *natural gradient* can be interpreted as the gradient of a function defined on the space of the probability distribution equipped with the Fisher metric. It can be also interpreted as the steepest ascent direction of the function with respect to the KL-divergence. Since the Fisher metric (and KL-divergence) is independent of the parameterization (coordinate system) of the probability distribution, the natural gradient is invariant to any re-parameterization of θ . Given a parameterization θ , the natural gradient is computed by the product of the inverse of the Fisher information matrix of θ and the vanilla gradient (gradient on the Euclidean space) of the log-likelihood of the probability distribution. We refer to [16] for further properties of the natural gradient.

Noting that $W_{\theta^t}^f$ in (2) or (3) is independent of θ , the natural gradient of J_{θ^t} is computed by

$$\tilde{\nabla} J_{\theta^t}(\theta) = \int_X W_{\theta^t}^f(x) \tilde{\nabla} l(\theta; x) P_{\theta^t}(dx) , \tag{4}$$

where $\tilde{\nabla}$ represents the map from a function to its natural gradient, and $l(\theta; x) = \ln p_\theta(x)$ denotes the log-likelihood at θ given x . Eq (4) is viewed as a weighted expectation of the natural gradient of the log-likelihood at θ .

Implementation of the Natural Gradient Ascent. The IGO algorithm performs the natural gradient ascent instead of exactly solving joint problem (1). Then iterate $\{\theta^t\}$ is defined by

$$\theta^{t+1} = \theta^t + \eta^t \tilde{\nabla} J_{\theta^t}(\theta)|_{\theta=\theta^t} , \tag{5}$$

where η^t denotes the step-size for the natural gradient ascent, aka the learning rate for the parameter update, which is sometimes replaced with a diagonal matrix whose diagonal entries are the learning rates for each element of the parameter vector. However, the integration in (4) cannot be performed analytically in advance unless f is known.

Therefore, we estimate (4) with samples x_1, \dots, x_λ drawn from P_{θ^t} . According to [16], we can approximate $W_{\theta^t}^f$ in (2) for each x_i as $W_{\theta^t}^f(x_i) \approx \hat{w}_{\text{rk}(x_i)} = w((\text{rk}(x_i) - 1/2)/\lambda)$, where $\text{rk}(x_i)$ denotes the ranking of $f(x_i)$ among $f(x_1), \dots, f(x_\lambda)$. With this, a Monte-Carlo estimate provides an approximation of (4) at $\theta = \theta^t$, namely

$$\tilde{\nabla} J_{\theta^t}(\theta)|_{\theta=\theta^t} \approx \frac{1}{\lambda} \sum_{i=1}^{\lambda} \hat{w}_{\text{rk}(x_i)} \tilde{\nabla} l(\theta^t; x_i) . \tag{6}$$

The IGO implementation performs the natural gradient ascent (5) with replacing the natural gradient $\tilde{\nabla} J_{\theta^t}(\theta)|_{\theta=\theta^t}$ given in (4) with its approximation (6).

Rank- μ Update CMA. Considering the IGO implementation for unconstrained continuous optimization, i.e. $X = \mathbb{R}^d$, with the Gaussian distributions on \mathbb{R}^d , it is known from [3] that the natural gradient of the log-likelihood of the Gaussian distribution is explicitly written in a special form. If the Gaussian distribution is parameterized by $\theta = [m^T, \text{vech}(C)^T]$, where m and C are the mean vector and the covariance matrix, the parameter update in the IGO implementation reads

$$\begin{aligned} m^{t+1} &= m^t + \frac{\eta m}{\lambda} \sum_{i=1}^{\lambda} \hat{w}_{\text{rk}(x_i)} (x_i - m) \\ C^{t+1} &= C^t + \frac{\eta c}{\lambda} \sum_{i=1}^{\lambda} \hat{w}_{\text{rk}(x_i)} ((x_i - m)(x_i - m)^T - C) . \end{aligned} \tag{7}$$

This is called the rank- μ update [10] and is a component of the standard CMA [9].

In [3], $X = \mathbb{R}^d$ is assumed to obtain the explicit form for the natural gradient. In other words, the natural gradient computed in the reference is the one on the manifold of (non-truncated) Gaussian distributions defined on \mathbb{R}^d . If X is a proper subset of \mathbb{R}^d ($X \subset \mathbb{R}^d$ and $X \neq \mathbb{R}^d$) and the truncated Gaussian distribution is considered (sampling from a Gaussian distribution with resampling scheme as a constraint handling), the natural gradient on the manifold of the truncated Gaussian distributions on X is different from the one derived in [3] and the resulting natural gradient ascent differs from (7). This is the main concern of the paper.

3 Natural Gradient for Truncated Gaussian Distributions

Let $p_{\theta}(x)$ and $l(\theta; x)$ be the probability density function (p.d.f.) and the log-likelihood function (l.l.f.) induced by the Gaussian distribution P_{θ} with mean $m = m(\theta)$ and covariance matrix $C = C(\theta)$, i.e., $l(\theta; x) = -\frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln \det(C(\theta)) - \frac{1}{2} (x - m(\theta))^T C^{-1}(\theta) (x - m(\theta))$ and $p_{\theta} = \exp(l(\theta; x))$. Then, $P_{\theta}(A) = \int_A p_{\theta}(x) dx$ for any Lebesgue measurable $A \subset \mathbb{R}^d$. As in the rank- μ update CMA-ES, we consider $\theta = [m^T, \text{vech}(C)^T]^T$.

If the X is a proper subset of \mathbb{R}^d and the resampling strategy is employed, the distribution of the samples in X is the Gaussian distribution truncated on X . Let $\bar{p}_{\theta}(x)$ and $\bar{l}(\theta; x)$ be the p.d.f. and l.l.f. of such a truncated Gaussian distribution \bar{P}_{θ} over X . Then, $\bar{p}_{\theta}(x) = p_{\theta}(x)/P_{\theta}(X)$ and $\bar{l}(\theta; x) = l(\theta; x) - \ln P_{\theta}(X)$ for $x \in X$, where $P_{\theta}(X) = \int_X p_{\theta}(x) dx = \mathbb{E}_{x \sim p_{\theta}}[\mathbb{I}\{x \in X\}]$ is the probability of x being sampled in X from p_{θ} . If we implement an algorithm following the IGO framework, the natural gradient on the manifold of the truncated Gaussian distributions $\{\bar{P}_{\theta} \mid \theta \in \Theta\}$ is needed.

As the first attempt of the work, we derive the natural gradient of the l.l.f. of the Gaussian distribution truncated on $X = \{x \in \mathbb{R}^d \mid \langle x, v \rangle \geq \alpha\}$. The following proposition and theorem provide the formula to compute the natural gradient explicitly.

Proposition 1. Let φ and Φ be the p.d.f. and cumulative density function induced by the standard normal distribution $\mathcal{N}(0, 1)$. Define φ_β be the p.d.f. for the normal distribution truncated onto $\{x \geq \beta\}$, that is, $\varphi_\beta(x) = \varphi(x)/(1 - \Phi(\beta))$. Let N_β be a random variable obeying φ_β . Then, $\mu_1 := \mathbb{E}[N_\beta] = \varphi_\beta(\beta)$, $\mu_2 := \mathbb{E}[N_\beta^2] = \beta\varphi_\beta(\beta) + 1$, $\mu_3 := \mathbb{E}[N_\beta^3] = (\beta^2 + 2)\varphi_\beta(\beta)$ and $\mu_4 := \mathbb{E}[N_\beta^4] = (\beta^3 + 3\beta)\varphi_\beta(\beta) + 3$.

Theorem 1. Let the natural gradient $\tilde{\nabla}l(\theta; x)$ of the l.l.f. of the truncated Gaussian distribution decomposed as $\tilde{\nabla}l(\theta; x) = [\delta m(x)^T, \text{vech}(\delta C(x))^T]^T$, where $\delta m(x) \in \mathbb{R}^d$ and $\delta C(x) \in \mathbb{R}^{d \times d}$ are the components corresponding to m and C respectively. Let $u = v/\|C^{1/2}v\|$ and define $y = x - m$. Then,

$$\delta m(x) = \left[\frac{\mu_2 - \mu_1 \langle u, y \rangle}{\tau_1} \right] y + \left[- \left(\frac{\tau_2}{\tau_1 \tau_3 - \tau_2^2} - \frac{\mu_1}{\tau_1} \right) \langle u, y \rangle^2 + \left(\frac{\tau_3}{\tau_1 \tau_3 - \tau_2^2} - \frac{\mu_2}{\tau_1} \right) \langle u, y \rangle - \frac{\tau_3 \mu_1 - \tau_2 \mu_2}{\tau_1 \tau_3 - \tau_2^2} \right] Cu \quad \text{and} \quad (8)$$

$$\delta C(x) = yy^T - C + \left[\frac{(1 - \tau_1) \langle u, y \rangle + \mu_1}{\tau_1} \right] (yu^T C + Cuy^T) + \left[\left(2 \frac{\tau_2 \mu_1 - \tau_1 \mu_2}{\tau_1 \tau_3 - \tau_2^2} + 1 \right) + \left(\frac{2\tau_1}{\tau_1 \tau_3 - \tau_2^2} - \frac{2}{\tau_1} + 1 \right) \langle u, y \rangle^2 - 2 \left(\frac{\tau_2}{\tau_1 \tau_3 - \tau_2^2} + \frac{\mu_1}{\tau_1} \right) \langle u, y \rangle \right] C u u^T C, \quad (9)$$

where μ_1, μ_2, μ_3 and μ_4 are as defined in Proposition 1 with $\beta = (\alpha - \langle v, m \rangle) / \|C^{1/2}v\|$, and $\tau_1 = \mu_2 - \mu_1^2, \tau_2 = \mu_3 - \mu_1\mu_2, \tau_3 = \mu_4 - \mu_2^2$.

We have omitted its proof due to the space limitation. Comparing to the natural gradient $\tilde{\nabla}l(\theta; x)$ of the l.l.f. for the non-truncated Gaussian distribution P_θ that can be expressed as $\delta m(x) = y$ and $\delta C(x) = yy^T - C$, (8) and (9) have additional components characterized by $Cu = Cv/\|C^{1/2}v\|$. The coefficients are determined by β —a signed distance to the boundary normalized by the standard deviation $\|C^{1/2}v\|$ in the direction of v —and $\langle u, y \rangle$ —a signed distance from the current mean m to the sample point x in the direction of $C^{1/2}v$. In the limit $\beta \rightarrow -\infty$, meaning that the constraint boundary is far away from the current mean and the situation is close to the unconstrained case, we have from Proposition 1 that $\mu_1 = \mu_3 = \tau_2 = 0, \mu_2 = \tau_1 = 1, \mu_4 = 3, \tau_3 = 2$, and (8) and (9) become identical to the natural gradient for the unconstrained case.

The natural gradient of the l.l.f. only depends on the manifold of the probability distributions. That is, it only depends on the feasible set X , but not on the objective function f . The parameter update (5) with (6) on the other hand depends on the selection scheme. More precisely, the adjustments δm and δC of the parameters is the weighted sum of $\delta m(x_i)$ and $\delta C(x_i)$ over $i = 1, \dots, \lambda$, where the weight value is determined by the ranking of $f(x_i)$. In the next section we demonstrate on a linearly constrained spherical problem how much the derived natural gradient differs from the rank- μ update (7).

4 Study on a Linearly Constrained Spherical Problem

We consider the following linearly constrained spherical problem $\text{argmin}_{x \in X_\alpha} f(x) := g(\|x\|^2)$, where $X_\alpha = \{x \in \mathbb{R}^d \mid \langle x, v \rangle \geq \alpha\}$ and g is strictly increasing. If $\alpha \geq 0$, the optimum is located on the boundary $x^* = \alpha v$, otherwise $x^* = (0, \dots, 0)$ and the landscape

around the optimum is the same as the unconstrained sphere function. Therefore, we consider only $\alpha \geq 0$ in this work.

For ranking-based weight scheme as in (2), the natural gradient (4) generally needs to be approximated by (6). To understand and emphasize the benefit of use the natural gradient derived in the previous section, we employ the weight scheme (3). As mentioned after (3), we can compute the joint objective analytically, $J_{\theta^f}(\theta) = -c(\|m\|^2 + \text{Tr}(C))$, on the unconstrained spherical problem and the natural gradients become $\overline{\delta m} = -2cCm$ and $\overline{\delta C} = -2cC^2$ with an appropriate constant factor c . This analytical natural gradient is the limit of the natural gradient estimate (6) w.r.t. $\lambda \rightarrow \infty$ [1] and it models the infinite-population behavior of the rank- μ update CMA (7).

If $\alpha = 0$, the volume of each sub level set $\mu_{\text{Leb}}[y \in X : f(y) \leq f(x)]$ is just halved compared to the unconstrained case and we still have similar results.

Lemma 1. *If $\alpha = 0$, the weight $W_{\theta^f}^f(x)$ defined in (3) is $-\tilde{c}x^T x$, where \tilde{c} is a constant independent of m and C . The natural gradient (4) with δm and δC derived in Theorem 1 reads $\overline{\delta m} = -2\tilde{c}Cm$ for m and $\overline{\delta C} = -2\tilde{c}C^2$ for C .*

Surprisingly, the natural gradient on the linearly constrained spherical problem is only different in length from the one on unconstrained spherical problem. This implies that the natural gradient update (5) with δm and δC in Theorem 1 reads the exact same parameter update as in the unconstrained case with an appropriate η^f . Therefore, all the results in [1] are carried over here as stated in the next theorem.

Theorem 2. *Let λ_1^t denote the largest eigenvalue of $-C^{-1/2}\overline{\delta C}C^{-1/2}$. If C^0 is symmetric positive definite and $\eta^f \lambda_1^t < 1$ for all $t \geq 0$, then C^t is symmetric positive definite. Moreover, if $\eta^f = \bar{\eta}/\lambda_1^t$ for $\bar{\eta} \in (0, 1/2]$, $\lim_{t \rightarrow \infty} \text{Cond}(C^t) = 1$ and $\lim_{t \rightarrow \infty} \|C^{t+1}\|_F / \|C^t\|_F = \lim_{t \rightarrow \infty} \|m^{t+1}\| / \|m^t\| = 1 - \bar{\eta}$, where $\|\cdot\|_F$ denotes the Frobenius norm.*

The learning rate $\eta^f = \bar{\eta}/\lambda_1^t$ is taken from [1]. This theorem means, the C becomes proportional to the Hessian of $x^T x$, namely the identity matrix, and m linearly converges towards the global optimum at the origin. For the detail, see [1].

To visualize the difference from the original parameter update (7) with $\lambda = \infty$ where the adjustment is $\mathbb{E}[W_{\theta^f}^f(x)(x - m)]$ and $\mathbb{E}[W_{\theta^f}^f(x)((x - m)(x - m)^T - C)]$, we derive the explicit form of the expectation. Following proposition reads it when the weight scheme with baseline subtraction, $W_{\theta^f}^f(x) - \mathbb{E}[W_{\theta^f}^f(x)]$, is introduced.

Proposition 2. *Let $y = x - m$, $u = v/\|C^{1/2}v\|$ and $\mu_1, \mu_2, \tau_1, \tau_2$ and τ_3 be as appeared in Theorem 1. If $\alpha = 0$, $\mathbb{E}[W_{\theta^f}^f(x)] = -\tilde{c}[\text{Tr}(C) + \|m\|^2 + 2\mu_1 u^T C m + (\mu_2 - 1)u^T C^2 u]$, $\mathbb{E}[y] = \mu_1 C u$, $\mathbb{E}[y y^T - C] = (\mu_2 - 1)C u u^T C$, and*

$$\begin{aligned} & \mathbb{E}[(W_{\theta^f}^f(x) - \mathbb{E}[W_{\theta^f}^f(x)])y] \\ &= -\tilde{c}[(\tau_2 - 2\mu_1)(u^T C^2 u) + 2(\tau_1 - 1)(u^T C m)]C u - 2\tilde{c}\mu_1 C^2 u - 2\tilde{c}C m \quad (10) \\ & \mathbb{E}[(W_{\theta^f}^f(x) - \mathbb{E}[W_{\theta^f}^f(x)])(y y^T - C)] \\ &= -2\tilde{c}C^2 - \tilde{c}[(\tau_3 - 4\mu_2 + 2)(u^T C^2 u) + 2(\tau_2 - 2\mu_1)(u^T C m)]C u u^T C \\ & \quad - 2\tilde{c}(\mu_2 - 1)(C^2 u u^T C + C u u^T C^2) - 2\tilde{c}\mu_1(C m u^T C + C u m^T C) . \quad (11) \end{aligned}$$

Note that the $\mathbb{E}[W_{\theta^f}^f(x)y] = \mathbb{E}[(W_{\theta^f}^f(x) - \mathbb{E}[W_{\theta^f}^f(x)])y] + \mathbb{E}[W_{\theta^f}^f(x)] \mathbb{E}[y]$ and $\mathbb{E}[W_{\theta^f}^f(x)(y y^T - C)] = \mathbb{E}[(W_{\theta^f}^f(x) - \mathbb{E}[W_{\theta^f}^f(x)])(y y^T - C)] + \mathbb{E}[W_{\theta^f}^f(x)] \mathbb{E}[y y^T - C]$. We call them NG_n

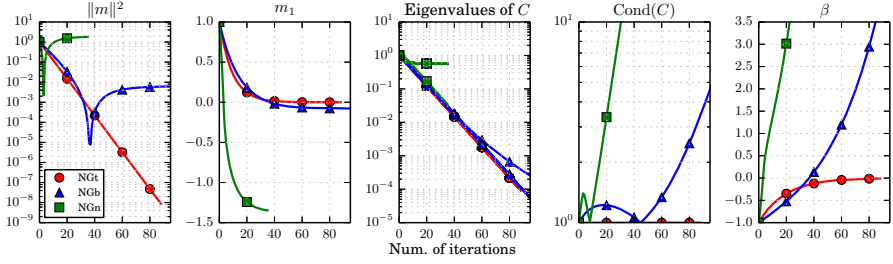


Fig. 1. Transitions of $\|m\|^2$, m_1 , the eigenvalues of C , the condition number $\text{Cond}(C)$, and β from left to right, respectively. For NG_n , the run stops after β reaches 7. In the center figure, the eigenvalue corresponds to the first coordinate is the larger one for NG_b , and the smaller one for NG_n .

(natural gradient computed on the non-truncated Gaussian manifold), in contrast to NG_t (natural gradient computed on the truncated Gaussian manifold) for $\bar{\delta}m$ and $\bar{\delta}C$ derived in Lemma 1. Moreover, we call (10) and (11) NG_b . The only difference between NG_n and NG_b is the offset of the weight, $-\mathbb{E}[W_{\theta'}^f(x)]$, and the expectation of the weight in NG_b is forced to be zero. Note that this offset does not affect the natural gradient in NG_t derived Lemma 1 since the expectations of $\delta m(x)$ and $\delta C(x)$ taken over x are zero.

Fig. 1 illustrates the evolution of the parameters m and C following the natural gradient update (5) with the natural gradient $\tilde{\nabla}J_{\theta'}(\theta) = [\bar{\delta}m^T, \text{vech}(\bar{\delta}C)^T]^T$ where $\bar{\delta}m$ and $\bar{\delta}C$ are computed for NG_t , NG_n , and NG_b . For the constraint we set $v = e_1 = [1, 0, \dots, 0]^T$ and $\alpha = 0$. The step-size is $\eta^t = \bar{\eta}/\lambda_1^t$, where $\bar{\eta} = 0.1$ and λ_1^t is the largest eigenvalue of $-C^{-1/2}\bar{\delta}C^{-1/2}$ with corresponding $\bar{\delta}C$ for each variant. This step-size setting guarantees the positivity of the covariance matrix as stated in Theorem 2 for NG_t . The problem dimension is $d = 10$. To produce simple figures, the evolution starts from $m^0 = e_1$ and $C^0 = I$. Thanks to the symmetry, m stays on the first axis and C stays to be a diagonal matrix whose second to the last diagonal elements are equal.

As stated in Theorem 2, m and C converge linearly in NG_t while the condition number of C stays 1 forever. In contrast, m goes over the constraint boundary and tends to stop at some point in the infeasible area while the condition number of C grows up in NG_b and NG_n . The normalized and signed distance β from m to the constraint boundary then becomes large, which in the actual algorithms means that the probability of sampling a point in the feasible region decreases. Since m does not converge towards the optimum, the best-so-far point would not converge linearly towards the global optimum. The tendency of the plots does not depend on the choice of the learning rate, i.e. β does not converge to zero with any learning rate in NG_n and NG_b .

So far the natural gradient is analytically computed. This is considered the approximated behavior of the algorithm in the limit of $\lambda \rightarrow \infty$. In practice the population size $\lambda < \infty$ and $W_{\theta'}^f(x)$ for each x_i and then $\bar{\delta}m$ and $\bar{\delta}C$ must be estimated using finite samples $x_1, \dots, x_{\lambda} \sim \bar{p}_{\theta}(x)$.¹ We can approximate $W_{\theta'}^f(x)$ as $\hat{W}_{\theta'}^f(x) :=$

¹ The resampling can be performed efficiently as follows. Generate $z \sim \mathcal{N}(0, I_d)$. If $\langle z, C^{1/2}u \rangle < \beta$, generate $\tilde{z} \sim \mathcal{N}(0, 1)$, resample it till $\tilde{z} \geq \beta$, then update $z = z + (\tilde{z} - \langle z, C^{1/2}u \rangle)C^{1/2}u$. Then, $m + C^{1/2}z$ obeys \bar{p}_{θ} . So we only need to resample a standard normal random number \tilde{z} .

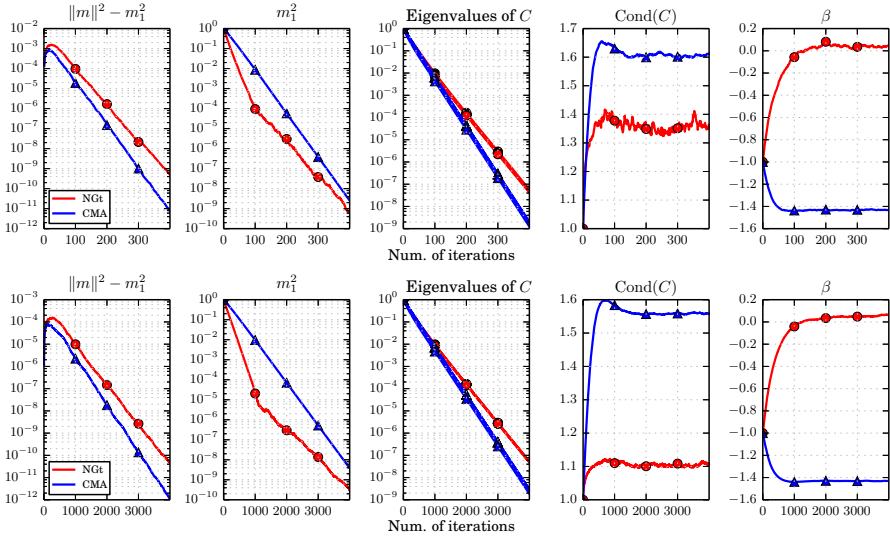


Fig. 2. Transitions of $\|m\|^2 - m_1^2$, m_1^2 , the eigenvalues of C , the condition number $\text{Cond}(C)$, and β for NG_t and the rank- μ update CMA with $\eta_m = \eta_c = \bar{\eta} = 0.1$ (above) and $\eta_m = \eta_c = \bar{\eta} = 0.01$ (below). The graphs are the average over 30 trials.

$[\sum_{j \in [k \in [1: \lambda] \|f(x_k) \leq f(x)]} (1/\bar{p}_\theta(x_j))]^{2/d}$, where $\bar{p}_\theta(x) = p_\theta(x)/P_\theta(X)$ and $P_\theta(X) = 1 - \Phi(\beta) = [1 - \text{erf}(\beta/\sqrt{2})]/2$.² With this approximation we can estimate the weight with baseline subtraction³, $W_\theta^f(x) - \mathbb{E}[W_\theta^f(x)]$, for each x_i as $w_i = \hat{W}_\theta^f(x_i) - \frac{1}{\lambda} \sum_{j=1}^\lambda \hat{W}_\theta^f(x_j)$. Then, $\overline{\delta m}$ and $\overline{\delta C}$ are approximated by the average of $w_i \cdot \delta m(x_i)$ and $w_i \cdot \delta C(x_i)$, respectively. We denote the estimated natural gradient by $\widehat{\delta m}$ and $\widehat{\delta C}$.

Fig. 2 compares the behaviors of NG_t with that of the rank- μ update CMA (7) (denoted CMA) with the weight used in the standard CMA, $\hat{w}_{\text{rk}(x_i)} = \max(0, \ln(\frac{\lambda+1}{2}) - \ln(\text{rk}(x_i)))/\sum_{j=1}^\lambda \max(0, \ln(\frac{\lambda+1}{2}) - \ln(j))$. The problem dimension $d = 10$, the population size $\lambda = 1000$, and the learning rates $\eta_m = \eta_c = 0.1$ and 0.01 for the rank- μ update CMA. For NG_t , $\eta^t = \bar{\eta}/\max[\sigma(C^{-1/2}\widehat{\delta C}C^{-1/2}), \|C^{-1/2}\widehat{\delta m}\|]$ with $\bar{\eta} = 0.1$ and 0.01 , where $\sigma(\cdot)$ represents the largest singular value. The graphs show the average of 30 independent runs for each method.

As you can see from the figure, in the rank- μ update CMA, β tends to stay at some point in negative, meaning that the mean vector is always away from the constraint boundary in the feasible domain and its distance is proportional to $\|C^{1/2}v\|$. The eigenvalue of C corresponding to $v = e_1$ becomes relatively smaller than the other eigenvalues.

² In practice, a scalar factor in \hat{W}_θ^f does not matter at all because the natural gradient is multiplied by η_m^t and η_c^t that are inversely proportional to the scalar factor as introduced below. Therefore, we can replace $1/\bar{p}_\theta(x)$ with $\exp(\|z\|^2/2)$, where $z = C^{-1/2}(x - m)$.

³ As stated above, the baseline subtraction in NG_t does not affect the expectation of the natural gradient, while it can reduce the estimation variance of the natural gradient.

On the other hand, NG_t results in smaller values of β and $\text{Cond}(C)$ and they get even smaller if we decrease the learning rate or increase the population size.

5 Summary and Discussion

In this paper we derive the natural gradient of the l.l.f. of the truncated Gaussian distribution for linearly constrained optimization problems. Analysis on a linearly constrained spherical problem shows the infinite-population model using the derived natural gradient reads the same update as the exact natural gradient algorithm on a unconstrained spherical problem [1] and all the results proven in the reference hold. The simulation results exhibit different behavior of the derived algorithm and the rank- μ update CMA. The rank- μ update CMA tends to stay in the feasible set and the distance from the constraint boundary stays proportional to $\|C^{1/2}v\|$, where v is the normal vector of the constraint boundary, whereas the condition number and the normalized distance in the derived algorithm converges to smaller values than in the rank- μ update CMA.

We would like to remark that the simulation performed in Section 4 depends heavily on the weight scheme. As we see in Fig. 1 and Fig. 2, NG_n , NG_b , and the rank- μ update CMA result in different behavior, although their only difference is the weight value and the learning rate. Moreover, from a preliminary experiment we have observed that NG_t does not work as well as it is with the weight scheme (3) if we employ the CMA-type weight scheme or the fitness proportional weight $\hat{W}_\theta^f(x_i) = \|x\|^2$. The mean vector enters the infeasible region as we have observed in NG_n . Especially for the fitness proportional weight, we can derive a theoretical result for the infinite-population model that even if $\alpha > 0$, the natural gradient becomes the same as the one on the unconstrained sphere problem and the mean vector tends to converges to the origin that is in the infeasible domain. Further study on the weight scheme is highly required.

For other future works, we compare the derived algorithm with the existing treatment for the constrained problem such as [6]. To enhance the performance, we would need to incorporate a step-size control mechanism that is in general heavily affected by the constraint, and a projection of the mean vector to the feasible domain when it reaches the infeasible domain. Furthermore, we extend the formula for the natural gradient for a linearly constrained problem stated in Theorem 1 to problems with more general constraint.

Acknowledgments. This work is supported by JSPS KAKENHI Grant Number 25880012.

References

1. Akimoto, Y.: Analysis of a Natural Gradient Algorithm on Monotonic Convex-Quadratic-Composite Functions. In: Genetic and Evolutionary Computation Conference, pp. 1293–1300 (2012)
2. Akimoto, Y., Nagata, Y., Ono, I., Kobayashi, S.: Bidirectional relation between CMA evolution strategies and natural evolution strategies. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 154–163. Springer, Heidelberg (2010)

3. Akimoto, Y., Nagata, Y., Ono, I., Kobayashi, S.: Theoretical Foundation for CMA-ES from Information Geometry Perspective. *Algorithmica* 64, 698–716 (2012)
4. Arnold, D.V.: Analysis of a repair mechanism for the $(1, \lambda)$ -ES applied to a simple constrained problem. In: Genetic and Evolutionary Computation Conference, pp. 853–860 (2011)
5. Arnold, D.V.: On the behaviour of the $(1, \lambda)$ -ES for a simple constrained problem. In: Foundations of Genetic Algorithms, pp. 15–24 (2011)
6. Arnold, D.V., Hansen, N.: A $(1 + 1)$ -CMA-ES for constrained optimisation. In: Genetic and Evolutionary Computation Conference, pp. 297–304 (2012)
7. Glasmachers, T., Schaul, T., Sun, Y., Wierstra, D., Schmidhuber, J.: Exponential Natural Evolution Strategies. In: Genetic and Evolutionary Computation Conference, pp. 393–400 (2010)
8. Hansen, N.: Benchmarking a BI-population CMA-ES on the BBOB-2009 noisy testbed. In: Companion on Genetic and Evolutionary Computation Conference (2009)
9. Hansen, N., Auger, A.: Principled Design of Continuous Stochastic Search: From Theory to Practice. In: Borenstein, Y., Moraglio, A. (eds.) *Theory and Principled Methods for the Design of Metaheuristics*. Springer (2013)
10. Hansen, N., Muller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* 11(1), 1–18 (2003)
11. Hansen, N., Niederberger, A.S.P., Guzzella, L., Koumoutsakos, P.: A Method for Handling Uncertainty in Evolutionary Optimization With an Application to Feedback Control of Combustion. *IEEE Transactions on Evolutionary Computation* 13(1), 180–197 (2009)
12. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
13. Harada, K., Sakuma, J., Ono, I., Kobayashi, S.: Constraint-handling method for multi-objective function optimization: Pareto descent repair operator. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) *EMO 2007*. LNCS, vol. 4403, pp. 156–170. Springer, Heidelberg (2007)
14. Harville, D.A.: *Matrix Algebra from a Statistician's Perspective*. Springer (2008)
15. Kramer, O.: A review of constraint-handling techniques for evolution strategies. *Applied Computational Intelligence and Soft Computing* 2010 (2010)
16. Ollivier, Y., Arnold, L., Auger, A., Hansen, N.: Information-geometric optimization algorithms: A unifying picture via invariance principles (2011), <http://arxiv.org/abs/1106.3708>
17. Rechenberg, I.: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog (1973)

Evolutionary Constrained Optimization for a Jupiter Capture

J eremie Labroqu ere*, Aur elie H eritier, Annalisa Riccardi, and Dario Izzo

Advanced Concepts Team, European Space Agency (ESA-ESTEC)
Noordwijk, The Netherlands

Abstract. This investigation considers the optimization of multiple gravity assist capture trajectories in the Jupiter system combining the well known Differential Evolution algorithm with different classes of constraint handling techniques. The trajectories are designed to reach a desired target orbit around Jupiter with minimum fuel consumption while satisfying mission design constraints on maximum thrust level, maximum time of flight and minimum closest distance to the planet. The advanced constraints handling techniques are compared for different set of constraints on the challenging mission design problem. For each method the trade off between performance, efficiency and the structure of the feasible space is analyzed in light of the results obtained.

1 Introduction

The exploration of planetary moons has become a scientific interest by space agencies such as NASA or ESA. The Jupiter system particularly has been the focus for recent mission concepts such as the JUICE mission [1]. These mission scenarios consist of a tour of Jupiter's moons using multiple flybys. One of their main goal is to assess the habitability of the four Galilean moons. Satellite-aided capture is a well-known trajectory design technique that is employed to decrease the fuel usage to capture a spacecraft into orbit around a planet.

Global optimization techniques have been successfully applied to interplanetary trajectory design [2,3]. They provide automated and unbiased searches for various trajectory options. Within the last decade, several researchers have investigated automated search techniques as a new approach to interplanetary trajectory design. Abdelklalik and Gad investigate genetic algorithms to determine both the optimal flyby sequence and the optimal trajectory [4]. Recently, Englander, Conway, and Williams develop an integer genetic algorithm to determine the optimal flyby sequence and employ differential evolution for the optimal trajectory [5]. However all these approaches are limited to the inclusion of constraints as penalty factors in the definition of the fitness function.

This current paper investigates an automated search procedure based on evolutionary techniques to design constrained interplanetary capture trajectories in

* Private work. Formally working at German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Lilienthalplatz 7, D-38108 Braunschweig, Germany.

the Jupiter system. The optimization problem is formulated as a constrained optimization problem. The trajectories are designed to target a final orbit around Jupiter with path constraints on the maximum acceleration and minimum distance to the center of the system. The system is evolved towards the minimization of the cumulative velocity increments. The purpose of this study is to investigate how different constraints handling schemes perform for different subsets of constraints, on the given trajectory optimization problem, and propose alternative techniques to the already widely used static penalty approach.

First, a modified version of the multiple gravity assist model (MGA-1DSM) for interplanetary design is presented for the formulation of the optimization problem [6]. The test case selected for this investigation is a capture trajectory in the Jupiter system using a predefined sequence of flybys at the Jupiter's moons and targeting a final orbit around Jupiter with an eccentricity constraint. The evolutionary optimization technique and the constraints handling methods selected for the study are described in the third section. The advantages and drawbacks of each method are briefly discussed while a quantitative comparative assessment, on the specific test case, is performed in the following section related to the experimental results. The summary of the results obtained and the future research directions are outlined as final conclusions of the paper.

2 Jupiter Capture Trajectory

The capture trajectory model is formulated as an optimization problem using a modified version of the MGA-1DSM model, where one deep-space maneuver is allowed between two successive flybys at the moons [7]. The initial conditions and characteristics of the spacecraft are taken from the problem statement of the Global Trajectory Optimization Competition (GTOC 6) that was organized by the Jet Propulsion Laboratory. Given a sequence of N moons, the constrained optimization problem is defined as

$$\begin{aligned}
 & \underset{x}{\text{minimize}} && \sum_{i=1}^N \Delta V_i \\
 & \text{subject to} && \text{lb} \leq x \leq \text{ub} \\
 & && a = a_{\text{final}}, e = e_{\text{final}}, i = i_{\text{final}} \\
 & && \Delta V_i < T_i a_{\text{max}}, i = 1, \dots, N \\
 & && \sum_{i=1}^N T_i < \text{tof}_{\text{max}} \\
 & && d_i > d_{\text{min}} \quad i = 2, \dots, N
 \end{aligned}$$

where the variable vector x is bounded between lower bounds and upper bounds, the objective function is composed of the sum of the deep space maneuvers and three types of inequality constraints and three equality constraints are considered. The first inequality constraint is an acceleration constraint that is applied

for each ΔV_i to meet the thrust requirements. The second one is a limit on the maximum time of flight and the third one is a minimum constraint on the closest approach to Jupiter to avoid damaging the spacecraft due to the high level of radiations emitted by the planet. The design of capture trajectories is often dictated by specific mission requirements related to the shape and orientation of the final insertion orbit. Therefore three equality constraints are introduced to define the desired semi-major axis, eccentricity and inclination of the final orbit. Given the set of moons defined as $\Gamma = \{I, E, C, G\}$ where $I = \text{Io}$, $E = \text{Europa}$, $G = \text{Ganymede}$ and $C = \text{Callisto}$, and consider a sequence of N moons, denoted Seq_N in Γ , i.e, $\text{Seq}_N \in \Gamma^N$. The objective is to find a x vector encoding an interplanetary trajectory that executes in sequence the N flybys at the moons and satisfies the constraints. Given a sequence of N moons, the variable vector has dimension $4N + 2$ and encodes the initial position, the flyby parameters, burn times and duration of each leg:

$$x = [t_0, u, v, T_0] + \sum_{i=1}^{N-1} [\beta_i, r_{p_i}/r_{\text{Planet}}, \eta_i, T_i] + [\beta_N, r_{p_N}/r_{\text{Planet}}]$$

The two last variables in the x vector denoted β_N and $r_{p_N}/r_{\text{Planet}}$ describing the last flyby are added to the traditional formulation of the MGA-1DSM model. These additional variables are necessary for the computation of the equality constraints associated with the shape of the final insertion orbit around Jupiter. The spacecraft is assumed to depart from a position located at $R_{\text{init}} = 1000 \text{ JR}$ from Jupiter center. Details on the problem statement and its mathematical formulation can be found in the problem description of the GTOC 6 competition.¹ The initial position of the spacecraft r_0 is described in spherical coordinates as $r_0 = R_{\text{init}}(\cos\theta\cos\hat{\phi}\hat{i} + \sin\theta\cos\hat{\phi}\hat{j} + \sin\hat{\phi}\hat{k})$ where the two angles, θ and $\hat{\phi}$ are defined as $\theta = 2\pi u$ and $\hat{\phi} = \arccos(2v - 1) - \pi/2$, respectively. The u and v variables are employed instead of θ and $\hat{\phi}$ to get a uniform distribution over the starting sphere of radius equal to 1000 JR. The launch date is represented by t_0 using the Modified Julian Date 2000 (MJD2000). The total duration of the first leg is given by T_0 . After reaching the first moon, the trajectory is propagated in a Keplerian model during $\eta_1 T_1$. A Lambert's solver is then employed to match the spacecraft position to the second moon in the sequence during $(1 - \eta_1 T_1)$. The flyby geometry at each moon is illustrated in Figure 1. The flyby is modeled as an hyperbolic path about the moons where the magnitude of the relative incoming hyperbolic velocity is equal to the magnitude of the relative outgoing hyperbolic velocity, i.e, $v_{\infty\text{-out}} = v_{\infty\text{-in}}$. The flyby angle δ describes how the spacecraft approaches the respective moon. More details on the flyby characteristics and the calculations of the b-plane angle β can be found in [8,5].

3 Constrained Evolutionary Optimization

Population based evolutionary techniques are all based on a common structure: a random set of solutions encoded into a chromosome is randomly initialized, then the

¹ http://sophia.estec.esa.int/gtoc_portal/

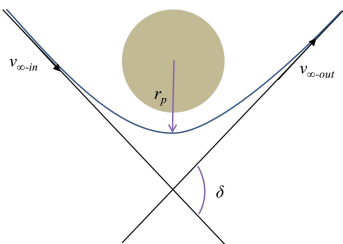


Fig. 1. Flyby geometry

chromosomes evolve through proper algorithm operators to create a new potential set of solutions. The process iterates from one solution set to another until a stopping condition is met. For each chromosome x a fitness function $F(x)$ is assigned as a measure of quality for the solution.

The self adaptive Differential Evolution (DE) algorithm [9] is an evolutionary technique that has already shown promising results in the design of interplanetary trajectory problems [7]. The parameters of the DE algorithm (variant $\text{rand}/1/\text{exp}$) [10,11], which represent the mutation parameter and the crossover constant, are encoded in the chromosome even though they do not evolve with the same operators. The possibility of having a dynamic updating rule for the algorithm parameters is particularly interesting in trajectory design problems, where the solutions in the early design phase are highly diversified.

3.1 Constraint Handling Techniques

DE and its self-adaptive variant jDE have been designed for single-objective unconstrained optimization. To solve for constrained single-objective optimization problems the evolutionary algorithm needs to be coupled with a constraint handling technique. Within the past few years several techniques have been developed to handle constrained optimization problems [12]. In this investigation a few of them have been selected based on their applicability and their different ways of dealing with the constraints.

Death Penalty. The rejection of the infeasible individuals is the most straightforward approach to handle constraints in evolutionary optimization. The fitness update rule of each individual has a penalty factor which become activated for infeasible individuals and assign to their fitness a large constant value. The process can easily get stuck if no feasible individual can be found. The Kuri variant of the method is considered [13], where the fitness function is updated as

$$F(x) = \begin{cases} f(x) & \text{if } x \text{ feasible} \\ K - \sum_{i=1}^s K/m & \text{otherwise} \end{cases}$$

where m is the number of constraints, s is the number of non violated constraints and K is a large constant.

Adaptive Penalty. More advanced adaptive penalty approaches have been developed to overcome the limitations of the previously presented techniques. In particular the co-evolution method proposed in [14] is considered for this study. It makes use of two populations P_1 and P_2 . The first one, P_1 , encodes the penalty coefficients while the second one, P_2 , encodes the optimization variables. The fitness of each individual belonging to P_2 is updated using the following fitness:

$$F(x) = f(x) + y_1 \sum_{i=1}^{m_I} \max[0, g_i(\mathbf{x})] + y_2 \text{Nvio}_{m_I} + y_3 \sum_{i=1}^{m_E} \max[0, h_i(\mathbf{x})] + y_4 \text{Nvio}_{m_E}$$

where (y_1^j, \dots, y_4^j) is the encoding of the j -th individual in P_1 , m_I and m_E are respectively the number of inequality and equality constraints $g(x)$ and $h(x)$, Nvio_{m_I} and Nvio_{m_E} are respectively the number of inequality and equality violated constraints. The fitness of each individual in the population P_1 depends on the entire population P_2 that is associated to it and it is a measure of its infeasibility. The two sets of populations are evolved cooperatively towards feasibility and optimality. The main drawback of such approach is its efficiency. The adopted formulation diverges from the original formulation of Coello Coello where equality and inequality constraints were aggregated in a single term.

Immune System. The technique emulates the biological behavior of an immune system making use of two populations and two optimization strategies [15]. In biological immune systems the antigenic molecules are recognized and then eliminated by the antibodies. Two populations are employed: one representing the antigens and one representing the antibodies. The fitness of an individual is determined by its ability to recognize antigens. Hence the population of antibodies, generated after the immune system simulation, is able to recognize antigens in the population and to evolve towards immunity and, therefore, feasibility.

Repair Methods. Repair methods are hybrid techniques that combine heuristic strategies with local searches. The infeasible individuals of the population are repaired to get closer to the feasible region. The repairing method considered in this investigation is a variation of the algorithm proposed in [16]. The infeasible individuals are repaired by means of a gradient descent algorithm that aims at minimizing the sum of the constraint violations. The outer optimization loop minimizes a penalized formulation of the original problem. This has been added to the original algorithm to preserve the repaired solutions reinjected into the population.

4 Experiments

The test case selected in this investigation considers 4 flybys in the Jupiter system. The predefined sequence of moons selected is $\text{Seq}_4 = \{C, G, G, G\}$. This capture

Table 1. Bounds on the variables vector **Table 2.** Bounds on the constraints for $i=0..3$ and $j=1..3$

Variables	lb	ub
t_0 [MJD 2000]	7305	11323
u [-]	0	1
v [-]	0	1
T_0 [days]	180	210
T_1 [days]	0.1	10
T_2 [days]	3	80
T_3 [days]	3	40
β_i ($i = 1..3$) [rad]	-2π	2π
r_{p_i}/r_{Planet} ($i = 1..3$) [-]	50	2000
η_i ($i = 1..3$) [-]	0	1
β_4 [rad]	-2π	2π
r_{p_4}/r_{Planet} [-]	50	2000

Constraints	lb	ub
$\Delta V_i/T_i$ [m/s^2]	$-\infty$	$5 \cdot 1e - 05$
$\sum_{i=0}^3 T_i$ [days]	$-\infty$	328.725
d_j [JR]	2	∞
e [-]	0.7-0.02	0.7+0.02

sequence corresponds to the one employed in previous work [7]. The values for the lower and upper bounds for the chromosome x , are defined in Table 1. The bounds of the inequality constraints introduced in Section 2 are reported in Table 2. The maximum acceleration is set by considering a thrust of 0.1 Newton and a spacecraft of 2000 kg, the maximum time of flight is constrained to 0.9 years and the minimum distance to the center of the system is constrained to 2 Jupiter radius. Concerning the equality constraints related to the shape of the final orbit, an eccentricity constraint is introduced as a proof of concept for this test case where $e_{final} = 0.7$. A high eccentricity is desirable, for example for the exploration of Jupiter and its environment. In particular the region between Callisto and Ganymede is interesting for magnetospheric/plasma physics science [17]. The implementation of the evolutionary and constraints handling techniques presented in Section 3 are made available as part of the open source scientific library PaGMO², and its python front-end PyGMO³. The framework also provides a generic interface to multiple well known optimization libraries. In particular, the local technique used in the experiments for the repair algorithm is the Nelder and Mead simplex algorithm from the GSL library⁴. The parameters involved in the constraints handling schemes have been tuned benchmarking a variety of constrained optimization problems taken from the 2006 IEEE Congress on Evolutionary Computation (CEC) competition, and have been kept the same in the different scenarios.

4.1 Problems Definition

Within gravity assist maneuvers, the desirable trajectories are the ones that do not require deep space maneuvers to reach a certain orbit. These specific trajectories are called *ballistic trajectories* and are defined such as $\sum_{i=0}^N \Delta V_i = 0$.

² <https://github.com/esa/pagmo/wiki>

³ <http://esa.github.io/pygmo/>

⁴ <http://www.gnu.org/software/gsl/>

These solutions, are very challenging to obtain within the feasible region because of the highly multimodal landscape of the fitness space. In the following, multiple problems definitions are considered with an increased complexity in terms of constraints satisfaction, see Table 3:

- **Case 1: Ballistic capture trajectory around Jupiter.** This case finds ballistic solutions under inequality constraints on the minimum distance to Jupiter and the maximum time of flight. The parameters of the final orbit around Jupiter, such as eccentricity, inclination and semi-major axis are free.
- **Case 2: Ballistic capture trajectory around Jupiter targeting a desired final eccentricity.** An extra equality constraint targeting a desired final eccentricity is added to the problem presented in case 1. The inclination and semi-major axis of the final orbit around Jupiter are free.
- **Case 3: Ballistic capture trajectory around Jupiter targeting a desired final eccentricity with constraints on maximum acceleration.** A maximum acceleration constraint on each trajectory leg is added to the problem definition of case 2.

Table 3. Details of the 3 test problems. $\rho = |F|/|S|$ is the estimated ratio between the feasible region and the search space, LI and NI are respectively the number of linear and nonlinear inequality constraints, LE and NE are respectively the number of linear and nonlinear equality constraints. a is the number of active constraints at optimality.

Problem	Constraints	ρ	LI	NI	LE	NE	a
Case 1	minimum distance to Jupiter, maximum time of flight	35.2097%	0	4	0	0	0
Case 2	minimum distance to Jupiter, maximum time of flight, final eccentricity	0.0000%	0	5	0	1	1
Case 3	minimum distance to Jupiter, maximum time of flight, final eccentricity, maximum acceleration	0.0000%	0	9	0	1	5

4.2 Results

All experiments have been sequentially run on a 1.8GHz i5 dual core processor. For each experiment, 250 runs are performed with a population size of 50 individuals. At each run, 400 evolutions for each algorithm are considered. Each algorithm contains 5000 internal generations to reach a maximum number of $1e8$ cost function evaluations. To keep the experiments computationally tractable within a reasonable time, the CPU time is limited to 2700s (45 minutes). The tolerances are set to $1e-8$ for both the cost function convergence and the algorithms internal tolerances. A special treatment is introduced for the case 3 by sequentially activating the acceleration constraints linked to each leg. This has been added because the constraint on the trajectory leg between Callisto and Ganymede is hard to be satisfied due to the limited time of flight allowed in the problem definition. Figure 2 illustrates the convergence of the different

constraints handling techniques towards ballistic solutions, with respect to the required CPU time. Table 4 reports the probability of finding a ballistic solution. Some observations on the results obtained can be stated:

- **Case 1:** The first test case shows comparable trend for the death penalty and repair techniques in terms of performance (probability of convergence) and efficiency (CPU time). The immune system has the best convergence performance but the worst convergence rate. Finally the co-evolution is the method that requires the highest CPU time to achieve convergence.
- **Case 2:** The behavior of the techniques is similar to the above case, except for the immune system that, even though it is able to reach feasibility, is not able to converge to the global optimum (ballistic solutions).
- **Case 3:** Only the co-evolution method is able to reach feasibility and global optimality. The required CPU time is comparable to the one of case 2.

The plot of a representative solution, for each of the test cases, is reported in Figure 3. For all constraints handling techniques, the solutions converge to an area of the search space where the orbits have very similar shapes. In order to achieve a greater variety in the set of final orbits, within ballistic solutions, also the sequence of moons needs to be enlarged and optimized.

To summarize, all the constraints handling techniques, but the immune system, were able to find feasible and ballistic solutions for the cases 1 and 2. For the case 3, only the co-evolution could find solutions. The repair method

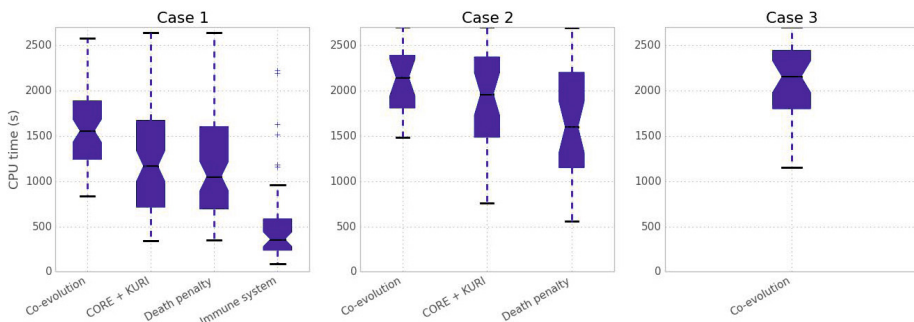


Fig. 2. Convergence performance, in terms of CPU time, towards ballistic solutions, of each constraints handling techniques

Table 4. Probability of convergence to a ballistic solution over 250 runs

Constraints handling technique	Convergence probability case 1	Convergence probability case 2	Convergence probability case 3
Co-evolution	0.272	0.1	0.152
CORE + Kuri	0.324	0.132	0
Death penalty	0.320	0.148	0
Immune system	0.192	0	0

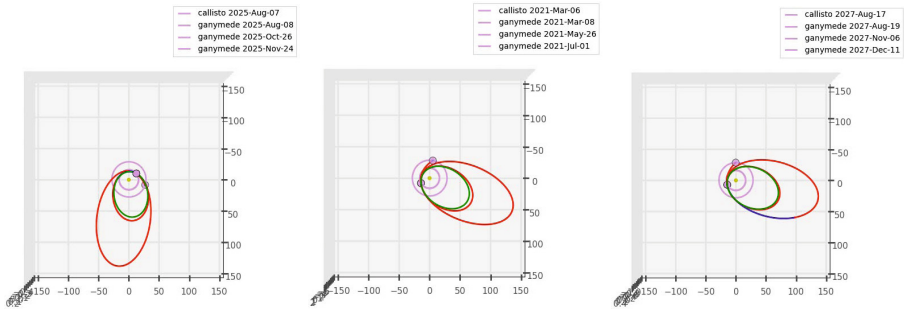


Fig. 3. Selected trajectories for the case 1, case 2 and case 3

CORE+Kuri and death penalty constraints handling technique are very similar in term of computational performance and probability of convergence for the first two cases, which highlights, that the repairing process does not have a large effect. Indeed if the constrained problem has a wide feasible region, easily reachable in the early stage of the evolution, the two techniques become comparable. They both fail in finding feasible solutions in the last test case. The adaptability of the penalty approach embedded in the co-evolutionary algorithm is the only strategy, between the selected ones, able to converge to feasibility and global optimality in each of the test case. As expected its main drawback, as illustrated in the first two cases, is its efficiency.

5 Conclusion and Prospects

In this paper an automatic procedure, using evolutionary constrained optimization techniques, for interplanetary trajectory design has been introduced. The trajectories are designed to reach a desired target orbit around Jupiter with minimum fuel consumption while satisfying mission design constraints on maximum thrust level, maximum time of flight and minimum closest distance to the planet. To optimize these trajectories, four constraints handling techniques have been introduced: the Kuri variant of the death penalty, the CORE+Kuri repairing method, the co-evolution and the immune system. All the techniques but the immune system could find feasible ballistic solutions within a respectable time. The immune system can't reach ballistic solutions as soon as the equality constraint on the final orbit eccentricity is targeted. The co-evolution technique is the only one able to find such solutions when maximum thrust level constraints are added to the problem. As a proof of concept only an eccentricity constraint is considered in this investigation. However as future work, additional equality constraints on semi-major axis and inclination can be included in the problem definition. Moreover another interesting aspect would be to optimize the sequence and number of moons to achieve a ballistic capture at Jupiter with insertion into any desired final orbit shape.

References

1. The JUICE Science Study Team: Juice exploring the emergence of habitable worlds around gas giants. ESA/SRE 18 (December 2011)
2. Izzo, D., Becerra, V.M., Myatt, D.R., Nasuto, S.J., Bishop, J.M.: Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. *Journal of Global Optimization* 38(2), 283–296 (2007)
3. Deb, K., Padhye, N., Neema, G.: Interplanetary trajectory optimization with swing-bys using evolutionary multi-objective optimization. In: Kang, L., Liu, Y., Zeng, S. (eds.) *ISICA 2007*. LNCS, vol. 4683, pp. 26–35. Springer, Heidelberg (2007)
4. Abdelkhalik, O., Gad, A.: Dynamic-size multi-population genetic optimization for multi-gravity-assist trajectories. *Journal of Guidance, Control, and Dynamics* 35(2), 520–529 (2012)
5. Englander, J.A., Conway, B.A., Williams, T.: Automated mission planning via evolutionary algorithms. *Journal of Guidance, Control, and Dynamics* 35(6) (2012), doi:10.2514/1.54101
6. Izzo, D.: Global optimization and space pruning for spacecraft trajectory design, spacecraft trajectory optimization, pp. 178–199. Cambridge University Press (2010)
7. Izzo, D., Simões, L.F., Märten, M., de Croon, G.C., Héritier, A., Yam, C.H.: Search for a grand tour of the jupiter galilean moons. In: *GECCO 2013: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, pp. 1301–1308 (2013)
8. Vinkó, T., Izzo, D.: Global optimisation heuristics and test problems for preliminary spacecraft trajectory design. ESA TR GOHTPPSTD (2008)
9. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation* 10(6), 646–657 (2006)
10. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11(4), 341–359 (1997)
11. Storn, R., Price, K.: Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical report (1995)
12. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering* 191(11-12), 1245–1287 (2002)
13. Morales, A.K., Quezada, C.V.: A universal eclectic genetic algorithm for constrained optimization. In: 6th Intelligent Techniques and Soft Computing European Congress, pp. 518–524 (1998)
14. Coello Coello, C.A.: Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry* 41(2), 113–127 (2000)
15. Hajela, P., Lee, J.: Constrained genetic search via schema adaptation: an immune network solution. *Structural Optimization* 12(1), 11–15 (1996)
16. Belur, S.V.: CORE: Constrained optimization by random evolution. In: Koza, J.R. (ed.) *Late Breaking Papers at the 1997 Genetic Programming Conference*, July 13-16, pp. 280–286. Stanford Bookstore, Stanford University (1997)
17. Campagnola, S., Kawakatsu, Y.: Jupiter magnetospheric orbiter trajectory design: Reaching high inclination in the jovian system. 22nd International Symposium on Space Flight Dynamics (February-March 2011)

Viability Principles for Constrained Optimization Using a (1+1)-CMA-ES

Andrea Maesani and Dario Floreano

Laboratory of Intelligent Systems,
Institute of Microengineering,
Ecole Polytechnique Fédérale de Lausanne (EPFL),
Lausanne, Switzerland
{andrea.maesani,dario.floreano}@epfl.ch
<http://lis.epfl.ch>

Abstract. Viability Evolution is an abstraction of artificial evolution which operates by eliminating candidate solutions that do not satisfy viability criteria. Viability criteria are defined as boundaries on the values of objectives and constraints of the problem being solved. By adapting these boundaries it is possible to drive the search towards desired regions of solution space, discovering optimal solutions or those satisfying a set of constraints. Although in previous work we demonstrated the feasibility of the approach by implementing it on a simple genetic algorithm, the method was clearly not competitive with the current evolutionary computation state-of-the-art. In this work, we test Viability Evolution principles on a modified (1+1)-CMA-ES for constrained optimization. The resulting method shows competitive performance when tested on eight unimodal problems.

Keywords: Stochastic optimisation, constrained optimisation, evolution strategy, viability evolution, constraint handling.

1 Introduction

Evolutionary computation methods are often used to solve real-valued black-box optimization problems, a large number of which require satisfying constraints. Without loss of generality, solving a real-valued constrained optimization problem in \mathbb{R}^n means minimizing the objective function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, subject to inequalities¹ defined on m constraints function $g_i(\mathbf{x}) \leq 0, i = 1, \dots, m$.

Several approaches have been proposed to solve constrained problems using evolutionary algorithms [1], ranging from rejecting solutions that violate constraints (infeasible solutions) to more sophisticated strategies that modify the ranking of individuals by penalizing the fitness using a function of constraint violations (penalty functions). Other popular approaches include stochastic ranking of solutions [2], ϵ -constrained optimization [3], feasibility rules to

¹ Equality constraints can always be rewritten as inequalities by using a tolerance value on the equality.

rank solutions [4], and transformation of constraints into objectives. Although these methods are necessary to handle infeasible solutions and constraints, an efficient optimizer is essential to progress during the search.

Currently, many state-of-the-art algorithms for unconstrained optimization are based on Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [5]. In CMA-ES, a covariance matrix describing correlations between decision variables is learned and adapted during the search to maximize the likelihood of generating successful solutions. Although CMA-ES is a powerful optimizer in unconstrained settings [6], it may suffer from premature convergence in presence of constraints, a common problem in strategies with adaptive step-size control [7]. Furthermore, methods for constrained optimization based on CMA-ES often require providing a feasible solution as a starting point.

A different modelling of objectives and constraints in CMA-ES may offer novel possibilities for handling constraints and allow the initialization of the algorithm from infeasible solutions. Viability Evolution [8,9] is an abstraction of artificial evolution that models an optimization process using viability boundaries, which are modified over time to drive the search towards desirable regions of a search space, as shown in Figure 1. Under this abstraction, mutations can produce viable solutions, which survive, or non-viable solutions, which are eliminated from the population. Viability boundaries are generally defined as admissible ranges of problem objectives and constraints. At the beginning of the search the boundaries are relaxed to encompass all randomly generated initial solutions and then gradually tightened. Once viability boundaries reach the desired target boundaries they are not tightened further, and the evolutionary process is considered complete.

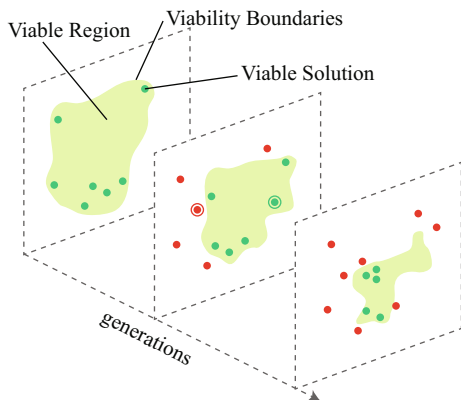


Fig. 1. Viability boundaries initially encompass all randomly generated solutions. We represent the viable region as a projection on a two-dimensional plane of the viability boundaries (shaded area). During the search, the boundaries are made more stringent. Viable solutions are retained in the population (dots in the shaded area), whereas solutions that do not satisfy viability boundaries are eliminated. Mutations can generate solutions (circled dots) that fall outside or inside the viable region.

In this work, we borrow concepts from Viability Evolution, and combine them with active covariance updates for CMA-ES [10], to derive a novel algorithm for constrained optimization. Here, we restrict ourselves to testing our method only in the case where it is started from a feasible solution, as done in [10] which reports current state-of-the-art performance on a set of eight unimodal functions.

The paper is structured as follows. In Section 2 we discuss the state-of-the-art of constraint handling in evolution strategies and we elucidate the workings of a (1+1)-CMA-ES with constraint handling proposed in [10]. In Section 3 we discuss Viability Evolution principles and the proposed approach for constrained optimization. Experimental setup and results of the proposed approach are presented in Section 4. Finally, we conclude with a brief discussion of the proposed approach in Section 5, and we propose future continuations of the work.

2 Related Work

Classical approaches to handle constraints in evolution strategies consist of simply discarding and resampling infeasible solutions [11] or using penalty functions. Penalty functions usually depend on the amount of constraints violation or number of violated constraints [12], and in some cases also on the fitness of selected feasible solutions [13]. The penalty functions can also be adaptive: for example the relative weight of each constraint in the penalty can be modified according to the number of iterations where infeasible solutions are discovered [14], or according to the ratio between feasible and infeasible individuals [15].

Other methods do not use penalty functions. An approach performs selection based on three feasibility rules [16]: feasible individuals are compared on objectives, infeasible ones are compared on total constraint violations, and feasible individuals are always ranked before infeasible ones. Similarly, a recently proposed method modifies the ranking of individuals based on three independent rankings: by objective function, by constraint violation amount, and by number of violated constraints depending on if the solution is feasible or infeasible [17]. Other approaches reduce the probability of generating infeasible solutions when in the proximity of the constraint, by moving the mean of the population [18] or by explicitly controlling the step size using a lower bound [7].

Another way in which constraints can be handled is learning surrogate models for linear constraints. One of these methods has been shown to be a promising approach to reduce the number of constraint function evaluations by predicting if solutions are feasible or infeasible, adapting directly the covariance matrix using the learned information, and repairing solutions that turn out to be infeasible [19]. The work has been recently extended to non-linear constraints, learning models using support vector machines [20]. Another recently proposed variant of CMA-ES [21] makes use of repair mechanisms, but the algorithm is very specific to the problem being solved (financial portfolio optimization).

2.1 (1+1)-CMA-ES with Active Covariance Matrix Adaptation

Among the various methods proposed for handling constraints in CMA-ES, Arnold and Hansen [10] recently proposed a modification of a (1+1)-CMA-ES

that has displayed great performance improvements with respect to other methods on unimodal constrained problems when started from a feasible solution. The method maintains a (low-pass filtered) vector representing the direction of violations of steps with respect to each constraint. These vectors are used to update the covariance matrix such that the variance in the direction of violation is reduced. A (1+1)-CMA-ES combines (1+1) selection [22] with covariance matrix adaptation [5]. Given a parent solution $\mathbf{x} \in \mathbb{R}^n$, an offspring solution \mathbf{y} is sampled according to $\mathbf{y} \leftarrow \mathbf{x} + \sigma \mathbf{A} \mathbf{z}$ where \mathbf{A} is the Choleski decomposition of the covariance matrix $\mathbf{C} = \mathbf{A}^T \mathbf{A}$ and $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is sampled from a normal distribution. The global step size $\sigma \in \mathbb{R}_+$ is changed according to a modified 1/5 rule proposed in [23]. The probability $P_{succ} \in [0, 1]$ of generating successful solutions and σ are updated at each iteration

$$P_{succ} \leftarrow (1 - c_p) P_{succ} + c_p \mathbb{1}_{f(\mathbf{y}) \leq f(\mathbf{x})} \tag{1}$$

$$\sigma \leftarrow \sigma \exp \left(\frac{1}{d} \left(P_{succ} - \frac{P_{target}}{1 - P_{target}} (1 - P_{succ}) \right) \right) \tag{2}$$

where $\mathbb{1}_{f(\mathbf{y}) \leq f(\mathbf{x})}$ is 1 if the condition is true or 0 otherwise, the learning rate $c_p \in (0, 1]$ determines the fading of P_{succ} and the damping factor d controls the step size variation. P_{target} determines the probability threshold that decreases or increases σ . The covariance matrix is adapted using the original rank-one update rule of CMA-ES, $\mathbf{C}^{(g+1)} = \alpha \mathbf{C}^{(g)} + \beta \mathbf{v}^{(g)} \mathbf{v}^{(g)T}$, which increases the variance in the direction of the provided vector \mathbf{v} from one iteration g to the following one. Using a vector of fading successful steps \mathbf{s} , called the evolution path, in place of vector \mathbf{v} , allows the strategy to increase the likelihood of sampling new solutions in the direction of already successful steps. In fact, there is no need to maintain the covariance matrix \mathbf{C} , as updates can be performed directly on the Choleski factor \mathbf{A} as proved in [23] according to

$$\mathbf{A} \leftarrow \sqrt{\alpha} \mathbf{A} + \frac{\sqrt{\alpha}}{\|\mathbf{w}\|^2} \left(\sqrt{1 + \frac{\beta}{\alpha} \|\mathbf{w}\|^2} - 1 \right) \mathbf{s} \mathbf{w}^T \tag{3}$$

where $\mathbf{w} = \mathbf{A}^{-1} \mathbf{s}$ and $\beta = c_{cov}^+ \in \mathbb{R}^n$. In practice the evolution path \mathbf{s} and α are updated depending on P_{succ} . If the probability of success is small ($P_{succ} < P_{thresh}$) then the covariance matrix is updated considering the current step $\mathbf{A} \mathbf{z}$, such that $\mathbf{s} \leftarrow (1 - c) \mathbf{s} + \sqrt{c(2 - c)} \mathbf{A} \mathbf{z}$ and $\alpha = 1 - c_{cov}^+$. Otherwise ($P_{succ} \geq P_{thresh}$), the update does not consider the current step in order to avoid the variance increasing too much in the direction of already successful mutations. In this case the covariance matrix is always updated using Equation 3 but the evolution path is set to $\mathbf{s} \leftarrow (1 - c) \mathbf{s}$ and $\alpha = 1 - c_{cov}^+ + c_{cov}^+ c(2 - c)$.

An alternative ‘‘active’’ covariance matrix update that also considers particularly unsuccessful steps worse than the fifth ancestor of the current solution was proposed in [24]. In this case, the covariance matrix is updated using the current

unsuccessful step $\mathbf{A}\mathbf{z}$ and the following rule that decreases the variance in the direction of that step²

$$\mathbf{A} \leftarrow \sqrt{\alpha}\mathbf{A} + \frac{\sqrt{\alpha}}{\|\mathbf{z}\|^2} \left(\sqrt{1 - \frac{\beta}{\alpha}\|\mathbf{z}\|^2} - 1 \right) \mathbf{A}\mathbf{z}\mathbf{z}^T \quad (4)$$

where $\alpha = \sqrt{1 + c_{cov}^-}$ and $\beta = c_{cov}^-$.

Interestingly, the same rule can be used to decrease variance in the direction of constraint violations. Similarly to what is done with the evolution path, Arnold and Hansen [10] proposed to use fading vectors of steps that violate constraints in combination with active covariance updates. Specifically, for each constraint i that is violated by step $\mathbf{A}\mathbf{z}$, the vector $\mathbf{v}_i \leftarrow (1 - c_c)\mathbf{v}_i + c_c\mathbf{A}\mathbf{z}$ is updated. Whenever even a single constraint is violated, the covariance matrix is updated according to

$$\mathbf{A} \leftarrow \mathbf{A} - \frac{B}{\sum_{i=1}^m \mathbb{1}_{g_i(\mathbf{y}) > 0}} \sum_{i=1}^m \mathbb{1}_{g_i(\mathbf{y}) > 0} \frac{\mathbf{v}_i \mathbf{w}_i^T}{\mathbf{w}_i \mathbf{w}_i^T} \quad (5)$$

where $\mathbf{w}_i = \mathbf{A}^{-1}\mathbf{v}_i$. The parameters used in the algorithm are set to the following [24]: $d = 1 + \frac{n}{2}$, $c = \frac{2}{n+2}$, $c_c = \frac{1}{n+2}$, $c_p = \frac{1}{12}$, $B = \frac{0.1}{n+2}$, $P_{target} = \frac{2}{11}$, $P_{thresh} = 0.44$, $c_{cov}^+ = \frac{2}{n^2+6}$, and $c_{cov}^- = \frac{0.4}{n^{1.6+1}}$. We will refer to this method in the following as (1+1)-acCMA-ES (active constrained CMA-ES).

3 Introducing Viability in CMA-ES

Modelling an evolutionary algorithm using the Viability Evolution abstraction offers novel possibilities. For example, in the case of constrained optimization viability boundaries can be defined to relax problem constraints at the beginning of the search, and be made more stringent over time to lead solutions into the feasible regions. The key idea proposed here is to use changing viability boundaries that define admissible regions of the search space (viable regions) in combination with the active covariance matrix updates proposed by Arnold and Hansen [10]. Active covariance updates are used to decrease the variance in the direction of boundary violations. As the boundaries defined on constraint functions values can be relaxed, the algorithm is compatible with infeasible starting solutions. On the other hand, whenever a viable solution is generated, the standard covariance matrix update rule of (1+1)-CMA-ES is employed to increase the variance in the direction that generated the viable solution. Because different boundaries may affect the global probability of generating viable solutions P_{succ} , we maintain a vector of probability of success \mathbf{p}_{succ} , that tracks which boundary is more likely to cause the generation of non viable solutions. As depicted in Figure 2A, when the covariance matrix is well adapted to a boundary, the probability of generating a new viable solution is greater or equal to 50%. Otherwise, when the probability of success is lower than 50% for at least one

² Note that the sign in the parenthesis is inverted. Furthermore, if $\|\mathbf{z}\|^2 \geq \frac{1+c_{cov}^-}{2c_{cov}^-}$ then $c_{cov}^- = \frac{1}{2\|\mathbf{z}\|^2-1}$.

Algorithm 1. (1+1)-VIE-CMA-ES pseudo-code. Problem objectives and constraints are modelled using the viability boundaries abstraction. Parameters $d, c, c_c, c_p, B, P_{target}, c_{cov}^+$ and c_{cov}^- are defined as in [24].

Require: $\sigma \in \mathbb{R}_+$ initial global step size

```

1:  $\alpha \leftarrow 1 - c_{cov}^+, \beta \leftarrow c_{ccov}^+, s \leftarrow 0$ 
2:  $\mathbf{A} \leftarrow \mathbf{I}$ 
3: for  $i = 1 \dots m + 1$  do
4:    $\mathbf{v}_i \leftarrow [0, \dots, 0]_{n \times 1}$   $\triangleright$  The last  $\mathbf{v}_i$  and  $b_i$  correspond to the objective
5: end for
6:  $\mathbf{b} \leftarrow [\max(0, g_1(\mathbf{x})), \dots, \max(0, g_m(\mathbf{x})), \infty]$ 
7:  $\mathbf{p}_{succ} \leftarrow [\frac{1}{2}, \dots, \frac{1}{2}]$ 
8:  $\mathbf{x} \leftarrow$  randomly generate solution
9: while  $\neg$  termination condition do
10:   $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ 
11:   $\mathbf{y} \leftarrow \mathbf{x} + \sigma \mathbf{A} \mathbf{z}$ 
12:   $\mathbf{V} \leftarrow [\mathbb{1}_{g_1(\mathbf{y}) > b_1}, \dots, \mathbb{1}_{g_m(\mathbf{y}) > b_m}, \mathbb{1}_{f(\mathbf{y}) > b_{m+1}}]$   $\triangleright$  Boundary violations
13:  if  $\exists i : V_i = 1$  then
14:    for all  $i : V_i = 1$  do
15:       $\mathbf{v}_i \leftarrow (1 - c_c)\mathbf{v}_i + c_c \mathbf{A} \mathbf{z}$ 
16:       $\mathbf{w}_i \leftarrow \mathbf{A}^{-1} \mathbf{v}_i$ 
17:    end for
18:     $\mathbf{A} \leftarrow \mathbf{A} - B \sum_{i=1}^m \mathbb{1}_{g_i(\mathbf{y}) > 0} \frac{\mathbf{v}_i \mathbf{w}_i^T}{\mathbf{w}_i \mathbf{w}_i^T}$   $\triangleright$  Decrease variance
19:     $\mathbf{p}_{succ} \leftarrow (1 - c_p)\mathbf{p}_{succ} + c_p [\mathbb{1}_{V_1=0}, \dots, \mathbb{1}_{V_{m+1}=0}]$   $\triangleright$  Update success
    probability
20:    if  $\exists i : \mathbf{p}_{succ_i} < \frac{1}{2}$  then
21:       $P_{succ} \leftarrow (1 - c_p)P_{succ}$   $\triangleright$  Decrease global  $P_{succ}$ 
22:    end if
23:  else
24:     $P_{succ} \leftarrow (1 - c_p)P_{succ} + c_p$   $\triangleright$  Increase success probabilities
25:     $\mathbf{p}_{succ} \leftarrow (1 - c_p)\mathbf{p}_{succ} + c_p$ 
26:     $\sigma \leftarrow \sigma \exp\left(\frac{1}{d} \left(P_{succ} - \frac{P_{target}}{1 - P_{target}}(1 - P_{succ})\right)\right)$ 
27:     $\mathbf{s} \leftarrow (1 - c)\mathbf{s} + \sqrt{c(2 - c)} \mathbf{A} \mathbf{z}$ 
28:     $\mathbf{w} \leftarrow \mathbf{A}^{-1} \mathbf{s}$ 
29:     $\mathbf{A} \leftarrow \sqrt{\alpha} \mathbf{A} + \frac{\sqrt{\alpha}}{\|\mathbf{w}\|^2} \left(\sqrt{1 + \frac{\beta}{\alpha} \|\mathbf{w}\|^2} - 1\right) \mathbf{s} \mathbf{w}^T$ 
30:     $\mathbf{b}_{1..m} \leftarrow \left[\max\left(0, \min\left(b_1, g_1(\mathbf{y}) + \frac{b_1 - g_1(\mathbf{y})}{2}\right)\right), \dots, \right.$ 
31:       $\left. \max\left(0, \min\left(b_m, g_m(\mathbf{y}) + \frac{b_m - g_m(\mathbf{y})}{2}\right)\right)\right]$ 
32:    if  $V_{i:1, \dots, m} = 0$  then  $\triangleright$  Update boundary on objective when feasible
33:       $b_{m+1} \leftarrow f(\mathbf{y}) + \frac{f(\mathbf{x}) - f(\mathbf{y})}{2}$ 
34:    end if
35:     $\mathbf{x} \leftarrow \mathbf{y}$ 
36:  end if
37: end while

```

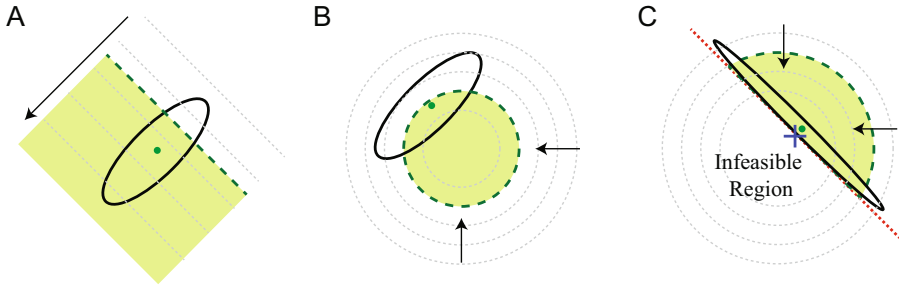


Fig. 2. Possible scenarios encountered during a search. A) The covariance matrix (ellipsoid in solid line) is well adapted with respect to a boundary (dashed line). The probability of generating a successful solution in the viability region (shaded area) is greater than 50%. Isocline of the objective function are shown as thin dotted lines and the gradient direction is shown by the arrow. The mean of the search distribution is represented as a dot. B) The covariance matrix should be adapted. Probability of generating successful solutions is lower than 50%. C) The method encounters difficulties when the direction to reach the optimum (shown as a cross) is the same that generates infeasible solutions that violate the constraint (thick dotted line).

boundary, as shown in Figure 2B, the covariance matrix should be modified and the global step size reduced. To achieve this, we reduce the global P_{succ} probability. Conversely, the overall P_{succ} probability and all elements of the \mathbf{p}_{succ} vector are increased whenever a viable solution is generated. Note that in the method presented in [10] not adapting P_{succ} on failure may lead to the use of outdated information for step-size adaptation.

The pseudo-code of our method, referred to as (1+1)-VIE-CMA-ES, is presented in Algorithm 1. The user must only provide an initial step size σ . The algorithm sets the initial viability boundaries b as either the target boundary (0 for the constraints) or a relaxed value if an infeasible solution is provided. The initial boundary for the objective is set to ∞ . At each iteration, boundary violations V are checked. The active covariance matrix update for feasible solutions (Equation 4) and the stall of updates of the original method in presence of high probability of success are not used. A single update rule is applied whenever a viable solution (that does not violates the boundaries b) is generated. When this happens, the mean of the population is updated to the new viable solution and the boundaries are tightened.

4 Results

The proposed method was tested on all the eight benchmark functions used in [10]. These benchmark functions include problems from two to ten dimensions with up to eight non-linear constraints. The experimental setup is identical to the one reported in [10], including the same number of repetitions, equivalent generation of initial solutions, the same termination condition, and the same parameter settings for the (1+1)-CMA-ES. For each benchmark function we counted the total number of objective function and constraints function evaluations. We tested the method starting it 99 times from different initial solutions, uniformly sampled from the

Table 1. Experimental results of the (1+1)-VIE-CMA-ES and comparison against the (1+1)-acCMA-ES proposed in [10]

	g06		g07		g09		g10	
	VIE-CMA	acCMA	VIE-CMA	acCMA	VIE-CMA	acCMA	VIE-CMA	acCMA
Function Evaluations								
10th	282	272	1578	1939	1305	1430	1387	2794
50th	333	308	1794	2211	1452	1674	1697	3976
90th	385	364	2049	2703	1595	2074	2554	5369
Constraint Evaluations								
10th	797	827	7184	10435	3474	3626	7360	15621
50th	900	1060	7545	11283	3660	4106	8295	18781
90th	986	1223	8032	12704	3913	5075	11322	23088

	TR2		2.40		2.41		HB	
	VIE-CMA	acCMA	VIE-CMA	acCMA	VIE-CMA	acCMA	VIE-CMA	acCMA
Function Evaluations								
10th	465	376	863	1326	820	1483	638	623
50th	520	443	1023	1990	954	2271	734	768
90th	561	510	1209	3326	1100	3581	841	1150
Constraint Evaluations								
10th	751	616	3166	4551	3183	5235	2659	2338
50th	812	708	3570	6994	3449	8108	2893	2912
90th	884	839	3899	11114	3801	12056	3185	3970

solution space until a feasible solution is found. Iterations needed to obtain the starting feasible solution are not counted in the results, as in [10].

Results are reported in Table 1. The method is competitive on seven out of eight problem. Our method has medians lower than what were reported by Arnold and Hansen [10] for constraint function calls by a factor of 0.15, 0.33, 0.11, 0.56, 0.49, 0.57 on g06, g07, g09, g10, 2.40, 2.41 respectively and almost identical performance on HB. In the linear constrained sphere function problem TR2, our method exceeds values reported by Arnold and Hansen [10] by a factor 0.15. In one problem, g06, our method, while being better on the overall number of constraint evaluations, performs slightly worse on number of objective function evaluations.

In our view, one of the reasons of decreased performance in the TR2 problem probably lies in the specific orientation of the constraint. From experimental investigation, we observed that the mean of the search distribution tends to align to the normal direction to the optimum (a situation similar to the one depicted in Figure 2C), which in this case is also the same direction that is most likely to violate the constraint. Probably, in cases like this one when the direction of constraint violation is very close to the direction of viable solutions generation, the covariance matrix update should be stalled, or the variance should be decreased along the other axis.

5 Discussion and Future Work

In this paper, we proposed (1+1)-VIE-CMA-ES, a method that combines viability boundaries and active covariance matrix updates in a (1+1)-CMA-ES. Our algorithm showed competitive performance with respect to state-of-the-art methods on all the benchmark problems except on the constrained sphere function problem TR2. Further investigations are needed to solve the lower performance experienced on TR2. Here, we tested the method only when starting from feasible initial solutions, but our algorithm is also compatible with infeasible starting solutions. In the future, we will proceed with a rigorous evaluation of the method when initialized from infeasible solutions. Also, more research will be needed for tackling multimodal problems using the approach presented here.

It is important to note that dealing with constraints and objectives using the same algorithmic framework allows one to readily extend the method to situations not directly manageable by standard CMA-ES. We anticipate that the coupling of changing viability boundaries and active covariance updates could also potentially be used in multi-objective optimization. For example, a “virtual” boundary may be learned on the Pareto front and made more stringent over time to push solutions towards the optimal Pareto front. The combined use of viability boundaries and active covariance updates might pave the way for a new class of powerful algorithms that can manage unconstrained, constrained and multi-objective problems under the same algorithmic scheme.

Acknowledgements. The authors would like to thank Joshua E. Auerbach and Giovanni Iacca for precious advice and useful comments on the manuscript. This research has been supported by the Swiss National Science Foundation, grant 200021_127143 and the FET-Open Grant 308943 within the 7th Framework Programme for Research of the European Commission.

References

1. Mezura-Montes, E., Coello Coello, C.A.: Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation* 1(4), 173–194 (2011)
2. Runarsson, T.P.: Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation* 4(3), 284–294 (2000)
3. Takahama, T., Sakai, S.: Constrained optimization by the ϵ constrained differential evolution with an archive and gradient-based mutation. In: *IEEE Congress on Evolutionary Computation (CEC 2010)*, pp. 1–9. IEEE Press (2010)
4. Deb, K.: An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* 186(2-4), 311–338 (2000)
5. 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)
6. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010)*, pp. 1689–1696. ACM Press (2010)
7. Kramer, O., Schwefel, H.: On three new approaches to handle constraints within evolution strategies. *Natural Computing* 5(4), 1–22 (2006)

8. Mattiussi, C., Floreano, D.: Viability Evolution: Elimination and Extinction in Evolutionary Computation. Technical Report (April 2003)
9. Maesani, A., Fernando, P.R., Floreano, D.: Artificial evolution by viability rather than competition. *Plos One* 9(1), e86831 (2014)
10. Arnold, D.V., Hansen, N.: A (1+1)-CMA-ES for constrained optimisation. In: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference (GECCO 2012), pp. 297–304. ACM Press (2012)
11. Schwefel, H.-P.P.: Evolution and optimum seeking: the sixth generation. John Wiley & Sons, Chichester (1993)
12. Hoffmeister, F., Sprave, J.: Problem-Independent Handling of Constraints by Use of Metric Penalty Functions. In: Proceedings of the Fifth Annual Conference on Evolutionary Programming, pp. 289–294. MIT Press (1996)
13. Oyman, A., Deb, K., Beyer, H.G.: An alternative constraint handling method for evolution strategies. In: Proceedings of the 1999 IEEE Congress on Evolutionary Computation, pp. 612–619. IEEE Press (1999)
14. Collange, G., Delattre, N., Hansen, N., Quinquis, I., Schoenauer, M.: Multidisciplinary Optimization in the Design of Future Space Launchers. In: Multidisciplinary Design Optimization in Computational Mechanics, pp. 487–496. John Wiley & Sons, Inc. (2010)
15. Kramer, O., Schlachter, U., Spreckels, V.: An adaptive penalty function with meta-modeling for constrained problems. In: IEEE Congress on Evolutionary Computation (CEC 2013), pp. 1350–1354. IEEE Press (June 2013)
16. Mezura-Montes, E., Coello Coello, C.A.: A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation* 9(1), 1–17 (2005)
17. Kusakci, A.O., Can, M.: A novel evolution strategy for constrained optimization in engineering design. In: XXIV International Conference on Information, Communication and Automation Technologies (ICAT), pp. 1–6. IEEE Press (October 2013)
18. Kramer, O., Ting, C.K., Büning, H.K.: A New Mutation Operator for Evolution Strategies for Constrained Problems. In: IEEE Congress on Evolutionary Computation (CEC 2005), vol. 3, pp. 2600–2606. IEEE Press (2005)
19. Kramer, O., Barthelmes, A., Rudolph, G.: Surrogate constraint functions for CMA evolution strategies. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 169–176. Springer, Heidelberg (2009)
20. Gieseke, F., Kramer, O.: Towards non-linear constraint estimation for expensive optimization. In: Esparcia-Alcázar, A.I. (ed.) EvoApplications 2013. LNCS, vol. 7835, pp. 459–468. Springer, Heidelberg (2013)
21. Beyer, H.G., Finck, S.: On the Design of Constraint Covariance Matrix Self-Adaptation Evolution Strategies Including a Cardinality Constraint. *IEEE Transactions on Evolutionary Computation* 16(4), 578–596 (2012)
22. Beyer, H.G., Schwefel, H.P.: Evolution strategies – a comprehensive introduction. *Natural Computing* 1(1), 3–52 (2002)
23. Igel, C., Suttrop, T., Hansen, N.: A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategies. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO 2006), pp. 453–460. ACM Press (2006)
24. Arnold, D.V., Hansen, N.: Active Covariance Matrix Adaptation for the (1+1)-CMA-ES. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010), pp. 285–392. ACM Press (2010)

On the Life-Long Learning Capabilities of a NELLI*: A Hyper-Heuristic Optimisation System

Emma Hart and Kevin Sim

Institute for Informatics and Digital Innovation, Edinburgh Napier University
Merchiston Campus, Edinburgh, EH10 5DT, UK
{e.hart,k.sim}@napier.ac.uk

Abstract. Real-world applications of optimisation techniques place more importance on finding approaches that result in acceptable quality solutions in a short time-frame and can provide robust solutions, capable of being modified in response to changes in the environment than seeking elusive global optima. We demonstrate that a hyper-heuristic approach NELLI* that takes inspiration from artificial immune systems is capable of life-long learning in an environment where problems are presented in a continuous stream and change over time. Experiments using 1370 bin-packing problems show excellent performance on unseen problems and that the system maintains memory, enabling it to exploit previously learnt heuristics to solve new problems with similar characteristics to ones solved in the past.

Keywords: Hyper-heuristics, artificial immune systems.

1 Introduction

Hyper-heuristics cover a general class of search methods that attempt to automate the process of selecting, combining, generating or adapting simple heuristics in order to solve large classes of problems. Although some compromise in solution quality is likely when comparing the quality of any single solution to a specifically tuned optimisation algorithm, the motivation is that this is compensated for by guaranteeing acceptable performance across very large problem sets, using cheap heuristics that are often simple to understand and can incorporate human knowledge.

Online hyper-heuristic methods [4] typically learn a sequence of low-level heuristics that can be applied to perturb an existing solution and learn during the solving phase. In contrast, *offline* methods attempt to find mapping between problem state and heuristic in order to determine how to solve a problem, requiring an initial offline training period using a representative set of problems (e.g. [18]). Both approaches potentially suffer from weaknesses. In the former, the hyper-heuristic learns from scratch each time a new instance of a problem is solved. In the latter, if the characteristics of the problem set change overtime, the hyper-heuristic needs to be periodically retuned. This potentially leads to inefficient algorithms that both fail to exploit previously learned knowledge in the search for a solution and cannot adapt to changing characteristics of problems.

Recently, Sim *et al* [13] proposed that the hyper-heuristic field could learn from new research in the machine-learning field in *moving beyond learning algorithms to more seriously consider the nature of systems that are capable of learning over a life-time*, stating that such systems must be capable of retaining knowledge (i.e. incorporate a memory) and of selectively applying that knowledge to new tasks. They described an approach inspired by Artificial Immune Systems (AIS) dubbed NELLI in which novel heuristics were continuously generated and a self-organising process determined whether they should be integrated into a self-sustaining network of problems and heuristics that could be used to solve a stream of problems continuously presented to the network. Results in [13] demonstrated that the system maintained memory, and was adaptable and efficient at solving problems when a dynamic stream of instances was presented. A modification to one of the key elements of the system — generating novel heuristics — was described in [15] (NELLI*) but was only demonstrated on static sets of problems in which the nature of the instances did not vary over time. Here, we demonstrate that using the new representation, not only is the system capable of life-long-learning but also that significant improvement is found over previous results when tested on a large corpus of bin-packing problems that vary in their characteristics and are presented in a continuous stream over time.

2 Background

In the hyper-heuristic domain, there are many examples of systems that either *generate* novel heuristics or *select* from a set of pre-defined heuristics to solve a problem and are shown to be capable of good performance across large problem sets, see [1] for a recent and comprehensive overview. However there is relatively little work that combines both generation and selection (recent examples include [7,9,17]). Additionally, although there is a wealth of work within the optimisation field addressing dynamic optimisation in which the fitness function applied to a single problem instance changes over time (e.g. [19]), we are not aware of other work that tackles problems in which the fitness function remains static but the characteristics of the instances presented varies over time.

In contrast, in the AIS literature, there are examples of systems that exhibit lifelong learning in response to changing environments, in particular in robotics. Typically, ‘antibodies’ specifying possible actions are connected in a plastic network that varies both in its topology and its constitution over time and determines an appropriate action to execute. This is typified by Whitbrook [20] who describes scenarios exhibiting both memory and adaptation in a robotics application — work which provided inspiration for NELLI*.

3 NELLI* Algorithm

Described in detail in [13] and visualised in Figure 1, the first version of NELLI comprised of three main parts: a stream of problem instances, a continuously generated stream of novel heuristics and a network that sustains co-stimulating components (heuristics and problem instances). NELLI is designed to run continuously; problem instances and heuristics can be added in any quantity at

any point. An AIS is responsible for governing the dynamic processes that enable heuristics and problem instances to be incorporated (stimulated) or rejected (suppressed) by the network. The original version ([13]) exploited a representation borrowed from Single Node Genetic Programming (SNGP) [6] in which a heuristic was represented by a tree randomly constructed from a set of terminal nodes that encapsulated information about the problem state (e.g. in the bin-packing domain, the free space in the bin and item size) and simple function nodes. All heuristics were randomly generated in this manner. In [15] two improvements were made. Firstly, the tree representation was replaced by a linear sequence of heuristic-components, each of which explicitly results in some items to be placed into the solution; secondly, rather than randomly generate all heuristics, a proportion p_m were generated by applying one of five mutation operators to an existing heuristic randomly chosen from the sustained network. By varying the value of p_m , it is possible to alter the balance between *exploration* (random generation of heuristics) and *exploitation* (focus the search around existing heuristics). The exploitation process is achieved through five operators:

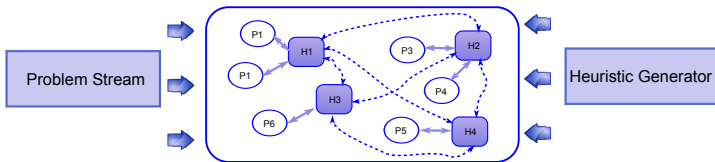


Fig. 1. A conceptual view of the system: Problems and heuristics are continuously injected into the AIS. The dynamics and meta-dynamics of the system result in a self-sustaining network of heuristics and problems. Heuristics and problems that receive no stimulation are removed.

- Select a random heuristic and swap the position of two random nodes.
- Select a random heuristic and replace a random node with a randomly selected node.
- Select a random heuristic from the network and remove a random node.
- Select a random heuristic and add a random node at a random position.
- Select two random heuristics from the network and concatenate their nodes.

Figure 2 shows a generic example of a heuristic represented by a string of five heuristic components with a “pointer” used by an encompassing wrapper to indicate the current component position. Each component is chosen from the list of nodes shown. If a node is successful in packing one or more items into a bin, then the pointer is advanced to the next node and the process continues with the current bin – when a node fails, a new bin is opened, and the pointer advances. The pointer is returned to the start after evaluation of the last node in the sequence. Each heuristic contains a sequence of nodes that are instantiated randomly up to a maximum *initial* length, specified by a parameter of the algorithm (l_{max}). The sequence may alter in length during the course of a run — the only constraint imposed is that the sequence must retain at least one node.

The network sustains a network of heuristics and problems through a process that varies the concentration of each element based on its *stimulation*. Problems

are *directly* stimulated by heuristics, and vice versa. Heuristics are *indirectly* stimulated by other heuristics. The total stimulation of a heuristic is the sum of its affinity with each problem in the set \mathcal{P} currently in the network \mathcal{N} . A heuristic h has a non-zero affinity with a problem $p \in \mathcal{P}$ *if and only if* it provides a solution that uses fewer bins than any other heuristic currently in \mathcal{H} . If this is the case, then the value of the affinity $p \leftrightarrow h$ is equal to the improvement in the number of bins used by h compared to the next-best heuristic. If a heuristic provides the best solution for a problem p but one or more other heuristics give an equal result, then the affinity between problem p and the heuristic h is zero. If a heuristic h uses more bins than another heuristic on the problem, then the affinity between problem p and the heuristic h is also zero. This is shown mathematically in Equations 1 and 2 described in detail in [13]. Essentially, the equations favour *heuristics* that are able to find a niche in solving at least one problem better than any other heuristic in the system, and *problems* that represent niche regions of the instance space, i.e. two or more heuristics do not perform equally on them.

$$h_{stim} = \sum_{p \in \mathcal{P}} \delta_{bins} \begin{cases} \delta_{bins} = \min(bins_{\mathcal{H}'_p}) - bins_{h_p} & : \text{if } \min(bins_{\mathcal{H}'_p}) - bins_{h_p} > 0 \\ \delta_{bins} = 0 & : \text{otherwise} \end{cases} \tag{1}$$

$$p_{stim} = \sum_{h \in \mathcal{H}} \delta_{bins} \begin{cases} \delta_{bins} = \min(bins_{\mathcal{H}'_p}) - bins_{h_p} & : \text{if } \min(bins_{\mathcal{H}'_p}) - bins_{h_p} > 0 \\ \delta_{bins} = 0 & : \text{otherwise} \end{cases} \tag{2}$$

Algorithm 1. NELLI* Pseudo Code

Require: $\mathcal{H} = \emptyset$:The set of heuristics

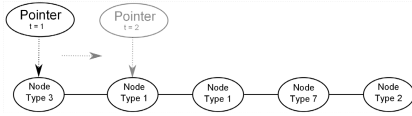
Require: $\mathcal{P} = \emptyset$:The set of current problems

Require: $\mathcal{E} = \mathcal{E}_{t=0}$:The set of problems to be solved at time t

1. **repeat**
 2. *optionally* replace $\mathcal{E} : \mathcal{E}^* \leftarrow \mathcal{E}^* \cup \mathcal{E}$
 3. **repeat**
 4. With probability p_m generate a new heuristic via mutation
 5. With probability $1 - p_m$ generate a new heuristic via random initialisation
 6. **until** n_h new heuristics generated
 7. Add n_h new heuristics to \mathcal{H} with concentration c_{init}
 8. Add n_p randomly selected problem instances from \mathcal{E} to \mathcal{P} with concentration c_{init}
 9. calculate $h_{stim} \forall h \in \mathcal{H}$ using Equation 1
 10. calculate $p_{stim} \forall p \in \mathcal{P}$ using Equation 2
 11. increment all concentrations (both \mathcal{H} and \mathcal{P}) that have concentration $< c_{max}$ and stimulation > 0 by Δ_c
 12. decrement all concentrations (both \mathcal{H} and \mathcal{P}) with stimulation ≤ 0 by Δ_c
 13. Remove heuristics and problems with concentration ≤ 0
 14. **until** *stopping criteria met*
-

4 NELLI* as a Life-Long-Learning System

In [15] we demonstrated that the linear representation described above improved the performance of NELLI in terms of finding optimal solutions when applied to a large but static set of problems. However, the capabilities of NELLI* as a life-long learning system were not demonstrated. The goal of the paper is to address this, showing that NELLI* is able to



Node Type	Description
1	Packs the single largest item into the current bin
2	Packs the largest combination of exactly 2 items into the current bin
3	Works as for 1 but packs exactly 3 items
4	Works as for 2 but packs exactly 4 items
5	Works as for 2 but packs exactly 5 items
6	Packs the largest combination of up to 2 items into the current bin giving preference to sets of lower cardinality.
7	As for 5 but considers sets of up to 3 items
8	As for 5 but considers sets of up to 4 items
9	As for 5 but considers sets of up to 5 items

Fig. 2. Heuristics are represented as linear sequences of nodes. A pointer keeps track of which node to apply next. The sequence restarts from the beginning after the last node is processed.

1. Demonstrate memory by quickly (re)finding solutions to problems that were seen by the system in the past
2. Continue to learn by continuing to improve its performance on problems in the datasets
3. Generalise, by quickly finding good solutions to problems that are similar to instances seen previously

We demonstrate this using a set of 1370 bin-packing problems taken from a variety of sources in the literature. Data sets $ds1$, $ds2$ & $ds3$ were introduced by [11] and comprise 720, 480 and 10 of the instances respectively. Problems in $ds1$, $ds3$ have optimal solutions with on average 3 items per bin and are similar in nature; solutions for problems in $ds2$ which have widely variable item weights have between 3 and 9 items per bin. Literature indicates that for a given algorithm, performance varies greatly on $ds2$ when compared to ($ds1$, $ds3$) [5]. The remaining instances are taken from $FalU$, $FalT$, and were introduced by [3]. All 1370 problems have known optimal solutions. In the following discussion we distinguish the following :

- \mathcal{U} - the set of 1370 problems from the class of 1D-BPP of interest.
- \mathcal{E} - the current environment, i.e. the set of problems we are currently interested in solving, $\mathcal{E} \subset \mathcal{U}$
- \mathcal{E}^* - the set of problems presented to the network so far
- \mathcal{P} - the set of problems currently sustained in the immune network $\mathcal{P} \subset \mathcal{E}^*$ (this is an internal property of the network)

5 Results

Four experiments were conducted using parameters described in Table 1 (taken from [15] where the tuning process is described). Performance was evaluated in terms of the number of problems solved optimally and in terms of the number of bins required over the known optimal solutions.

Clearly the problems considered have been solved by a plethora of optimisation methods in the past (including hyper-heuristics). For instance, Burke *et al* [2] obtain excellent results on the 90 problem instances in *FalU* and *ds3* by evolving an individual heuristic per problem instance using 50000 evaluations for each instance. Others seek optimality using exact methods (e.g. [11]) to solve each instance separately. In contrast, NELLI aims to find high quality solutions to very large sets of problems by only evaluating a very small subset of the instance space, therefore, direct comparisons with ‘per-instance’ methods cannot be made. Some earlier hyper-heuristic methods do attempt a similar kind of generalisation (e.g. [10]), however as we have already shown in [13] that the original version of NELLI outperforms these methods we omit these comparisons.

In addition to comparing to NELLI [13], we compare our results to those obtained by a set of four well known heuristics from the literature (FFD, DJD, ADJD and DJT, see [16]) in which the best heuristic for each problem is selected using a greedy approach. Additionally, where appropriate we also compare to our own earlier work using an AIS model introduced in [17] and an island-model evolutionary algorithm described in [14].

Table 1. Default parameter settings for experiments

Parameter	Description	Value
n_p	number of problems added each iteration	30
n_h	number of heuristics added each iteration	1
c_{init}	initial concentration of heuristics/problems	200
Δ_c	variation in concentration based on stimulation	50
c_{max}	maximum concentration level	1000
p_m	Probability of mutation	0.75
l_{max}	maximum initial heuristic sequence length	10

5.1 Generalisation Capabilities

In order to test the generalisation capabilities of NELLI*, in the first instance the full set of 1370 instances was randomly split into two equally sized sets, labelled *train* and *test* — each set contained an equal distribution of examples from each of the 5 constituent problem sets. NELLI* was run for 200 iterations using the *train* set before being presented with the unseen problems in the *test* set. Performance was evaluated in terms of the number of problems solved optimally and in terms of the number of bins required over the known optimal solution and is shown in table 2 where all results are averaged over 20 runs.

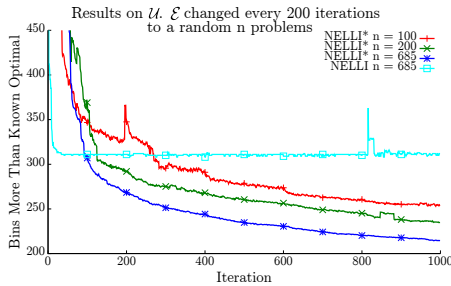
NELLI* clearly generalises, finding the optimal solution to 82.6% of the unseen problems, and reducing the number of bins over optimal in comparison to the other algorithms. Comparing the results obtained using NELLI and NELLI* using a t-test proves the result is highly significant ($P < 0.0001$). Although not shown, NELLI* continues to learn and improve results (albeit at a slower rate) if executed over a much larger number of generations.

An additional experiment was performed in which every 200 iterations, a random set of n instances were selected from \mathcal{U} to form the environment \mathcal{E} and presented to the system. At each iteration, the performance of the system against \mathcal{U} (the complete universe of 1370 problems) is measured. Recall that

Table 2. Results on the unseen 685 problems in the test set after 200 iterations training on the 685 problems in the training set

	Problems Solved				Extra Bins			
	min	max	mean	sd	min	max	mean	sd
Greedy Heuristic selection	548	548	548	0	188	188	188	0
AIS model [17]	554	559	556	1.4	159	165	162	1.4
Island Model [14]	552	559	557	1.4	159	164	162	1.4
NELLI [13]	559	559	559	0	159	159	159	0
NELLI*	549	576	566	5.8	131	164	146	8.2

at each iteration, the environment contains at most $n = 685$ problems, and only $n_p = 30$ of these are presented to the network at each instance, hence many of the instances in \mathcal{U} have never been seen. Figure 3 plots performance over \mathcal{U} over 1000 iterations (averaged over 20 runs) for $n \in 100, 200, 685$ and compares the results to the best result obtained by the old version of NELLI. A number of important points are apparent: NELLI* clearly outperforms NELLI; it generalises over \mathcal{U} — recall that in the early iterations most of the problems in \mathcal{U} are unseen; it continues to learn over time; the ability to generalise is maximised by increasing the size of \mathcal{E} .

**Fig. 3.** The average number of bins greater than the optimal; the environment \mathcal{E} is replaced with n randomly selected problem instances every 200 iterations

5.2 Memory and Learning

In order to investigate the memory capabilities of NELLI* we conduct an experiment in which the environment \mathcal{E} is toggled between two different datasets every 200 iterations. The first dataset contains the 720 problems in $ds1$ and the second the 480 problem instances in $ds2$. As previously mentioned, these datasets have different characteristics such that heuristics that perform well on one dataset are not expected to perform well on the second. Typically, $ds1$ problems are also easier to solve. Each dataset is presented twice in the sequence $ds2, ds1, ds2, ds1$ following a ‘start-up’ epoch in which the system is initialised from scratch with $ds1$ and run for 200 iterations. For each dataset, we record the number of bins

more than optimal at the start of each epoch it is introduced (b_s), and at the end of each epoch (b_e). We formulate the following hypotheses:

- *Hypothesis 1* if the system has retained some memory of heuristics that previously solved these problems, we expect the value of b_e at epoch t to be similar to b_s at the next epoch the dataset is introduced (epoch $t + 2$)
- *Hypothesis 2* If the system continues to learn over an epoch, there *should* be a significant difference between b_s and b_e over an epoch during which the dataset is present
- *Hypothesis 3* If the system continues to learn over its lifetime, there *should* be a significant difference between b_e at the first epoch the dataset appears, and b_e at the last epoch it occurs

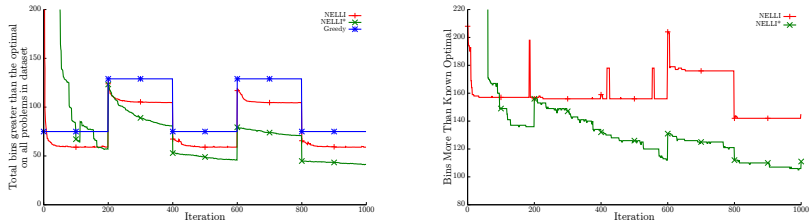
The results are shown in Table 3 and graphically in figure 4a, averaged over 20 runs in each case and compared to the results previously published in [17]. With respect to *hypothesis 1*, t-tests conducted on the values obtained at the end of epoch 1 and the start of epoch 3 for *ds2*, and epochs 2 and 4 for *ds1* show no significant difference between results ($P=0.581,0.581$), thus we infer that the system has retained memory¹. With respect to *hypothesis 2*, t-tests between the values of b_s and b_e at the two epochs where *ds2* appears both given p-values < 0.0001 , confirming that the observed difference in performance is significant. The same result is found for *ds1* at epochs 2 and 4. Thus, we confirm that learning occurs over an epoch. Finally, we compare the value of b_e at epoch 1 with b_e at epoch 3 ($P < 0.0001$) and similarly with epochs 2 and 4 ($P = 0.0138$) proving the ability of NELLI* to learn over time.

Figure 3 confirms these trends by further analysing the final experiment described in section 5.1 in which \mathcal{E} is changed to a randomly selected set of 685 instances from the 1370 problems in \mathcal{U} every 200 instances. The same general trends are observed as in Figure 4a in that learning continues across the 1000 iterations. The difference between b_e and b_s is less defined at each epoch in this instance, as problems in \mathcal{E} at epoch t are likely to overlap with those in \mathcal{E} at epoch $t + 1$. The magnitude of $|b_e - b_s|$ (where b_e is measured at the end of epoch t and b_s at the start of epoch $t + 1$) decreases over time, as a direct result of the *memory* of the system.

Table 3. Bins greater than the known optimal at epochs. Averaged over 20 runs

Set	Epoch 1 DS2		Epoch 2 DS1		Epoch 3 DS2		Epoch 4 DS1	
	Start	End	Start	End	Start	End	Start	End
NELLI* \mathcal{E}	123.28	80.61	52.94	45.78	79.39	70.72	44.83	41.28
NELLI \mathcal{E}	123.95	104.8	67.3	59	116.95	104.65	65.6	59
Greedy \mathcal{E}	129	129	75	75	129	129	75	75
NELLI* \mathcal{U}	312.89	259.17	257.67	247.22	246.89	233.94	234.44	229.17
NELLI \mathcal{U}	333.1	315.75	317.05	321.5	320.95	315.15	315.25	320.85
Greedy \mathcal{U}	364	364	364	364	364	364	364	364

¹ Strictly speaking, this only suggests that there is no evidence to suggest otherwise rather than providing proof.



(a) The average *total* number of bins greater than the optimal alternating every 200 iterations between ds1 (summed over 720 problems) and ds2 (summed over 480 problems).

(b) \mathcal{E} is changed every 200 iterations to a random 685 problems taken from \mathcal{U} . The graph shows the average *total* number of bins greater than the optimal summed over the 685 problems

Fig. 4. Performance on alternating datasets, averaged over 20 runs

6 Conclusions and Future Work

In an extension to previous work, we have shown that the NELLI* system is capable of operating as a life-long learning (LML) system. As identified by [12] in their recent proposal, the system exhibits the three defining characteristics of an LML system: it incorporates a long-term memory; it selectively transfers prior knowledge when learning new tasks; it adopts a systems approach that ensures the effective and efficient interaction of the elements of the system. In comparison to previous hyper-heuristic approaches, it obtains better performance when evaluated according to two metrics, number of problems solved, and number of bins over optimal. Although any hyper-heuristic method that focuses on solving large sets of problems will inevitably trade some loss in performance against both generality and speed when compared to approaches that optimise solutions for each problem individually, we believe that in practice, such solutions are more than acceptable. In a recent article considering why evolutionary algorithms are not widely adopted in the real-world [8], the author notes that in industry, organisations do not have time to generate globally optimum solutions and therefore place higher importance on finding robust, quality solutions that can be generated quickly due to changes in the environment. NELLI directly addresses this issue, in providing cheap, high quality solutions in a system that does not need either tuning or modifying even as the environment it operates in changes.

References

1. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. *J. Oper. Res. Soc.* (July 2013)
2. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.: Automating the packing heuristic design process with genetic programming. *Evol. Comput.* 20(1), 63–89 (2012)
3. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics* 2, 5–30 (1996)

4. Garrido, P., Riff, M.C.: Collaboration between hyperheuristics to solve strip-packing problems. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) IFSA 2007. LNCS (LNAI), vol. 4529, pp. 698–707. Springer, Heidelberg (2007)
5. Gent, I.P.: Heuristic solution of open bin packing problems. *Journal of Heuristics* 3(4), 299–304 (1998)
6. Jackson, D.: Single node genetic programming on problems with side effects. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 327–336. Springer, Heidelberg (2012)
7. Maturana, J., Lardeux, F., Saubion, F.: Autonomous operator management for evolutionary algorithms. *Journal of Heuristics* 16, 881–909 (2010)
8. Michalewicz, Z.: Ubiquity symposium: Evolutionary computation and the processes of life: The emperor is naked: Evolutionary algorithms for real-world applications. *Ubiquity 2012(November)*, 3:1–3:13 (2012)
9. Remde, S., Cowling, P., Dahal, K., Colledge, N., Selensky, E.: An empirical study of hyperheuristics for managing very large sets of low level heuristics. *J. Oper. Res. Soc.* 63(3), 392–405 (2012)
10. Ross, P., Schulenburg, S., Marin-Blazquez, 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 (2002)
11. Scholl, A., Klein, R., Jürgens, C.: Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Comput. Oper. Res.* 24(7), 627–645 (1997)
12. Silver, D., Yang, Q., Li, L.: Lifelong machine learning systems: Beyond learning algorithms. *AAAI Spring Symposium Series* (2013)
13. Sim, K., Hart, E., Paechter, B.: A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation Journal* (in press, January 2014)
14. Sim, K., Hart, E.: Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. In: Blum, C. (ed.) GECCO 2013: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, ACM, New York (2013)
15. Sim, K., Hart, E.: An improved immune inspired hyper-heuristic for combinatorial optimisation problems. In: GECCO 2014: Proceeding of the Sixteenth Annual Conference on Genetic and Evolutionary Computation Conference (in press, 2014)
16. Sim, K., Hart, E., Paechter, B.: A hyper-heuristic classifier for one dimensional bin packing problems: Improving classification accuracy by attribute evolution. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 348–357. Springer, Heidelberg (2012)
17. Sim, K., Hart, E., Paechter, B.: Learning to solve bin packing problems with an immune inspired hyper-heuristic. In: *Advances in Artificial Life, ECAL 2013: Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems*, pp. 856–863. MIT Press (2013)
18. Thabtah, F., Cowling, P.: Mining the data from a hyperheuristic approach using associative classification. *Expert Systems with Applications* 34(2), 1093–1101 (2008)
19. Trojanowski, K., Wierzchon, S.T.: Immune-based algorithms for dynamic optimization. *Information Sciences* 179(10), 1495–1515 (2009)
20. Whitbrook, A.M., Aickelin, U., Garibaldi, J.M.: Two-timescale learning using idiosyncratic behaviour mediation for a navigating mobile robot. *Appl. Soft Comput.* 10(3), 876–887 (2010)

Adaptation in Nonlinear Learning Models for Nonstationary Tasks

Wolfgang Konen and Patrick Koch

Department of Computer Science, Cologne University of Applied Sciences,
51643 Gummersbach, Germany
wolfgang.konen@fh-koeln.de

Abstract. The adaptation of individual learning rates is important for many learning tasks, particularly in the case of nonstationary learning environments. Sutton has presented with the Incremental Delta Bar Delta algorithm a versatile method for many tasks. However, this algorithm was formulated only for linear models. A straightforward generalization to nonlinear models is possible, but we show in this work that it poses some obstacles, namely the stability of the learning algorithm. We propose a new self-regulation of the model's activation which ensures stability. Our algorithm shows better performance than other approaches on a nonstationary benchmark task. Furthermore we show how to derive this algorithm from basic loss functions.

Keywords: Machine learning, IDBD, learning rates, adaptation.

1 Introduction

For many state-of-the-art learning algorithms the adaptation of learning rates (or other algorithm parameters) is important. This is particularly true if these algorithms shall behave well in nonstationary learning environments.

In 1992 Sutton [10] suggested the Incremental Delta Bar Delta (IDBD) algorithm. IDBD deals with the learning rates for trainable parameters of any underlying learning algorithm. The key idea of IDBD is that these learning rates are not predefined by the algorithm designer but they are themselves adapted as hyperparameters of the learning process. Sutton [10] expects such adaptable learning rates to be especially useful for nonstationary tasks or sequences of related tasks and demonstrates good results on a small synthetic nonstationary learning problem with 20 weights.

Sutton's algorithm is proposed to work with a linear model. But as many learning tasks exhibit nonlinear characteristics, e. g., well-known benchmark tasks for the control of physical objects like MountainCar or pole balancing, or real-world applications like the control of complex processes in plants. It is our goal to extend Sutton's IDBD to the nonlinear case. However, the nonlinear case poses some obstacles because the varying steepness in input-output relations can cause instabilities when IDBD is extended in a straightforward manner (by simply replacing

the linear input-output relation with the nonlinear one). We demonstrate these obstacles and describe a way to overcome them by using a weight decay method.

The paper is organized as follows: after briefly reviewing some related work in the next paragraph, we present the IDBD method and our nonlinear generalization n-IDBD in Sec. 2. In Sec. 3 we apply n-IDBD to a nonlinear, nonstationary benchmark task. We show in Appendix A how the equations of n-IDBD can be derived from basic loss functions.

1.1 Related Work

Several online learning rate adaptation schemes have been proposed over the years: IDBD [10] from Sutton is an extension of Jacobs' [4] earlier DBD (Delta Bar Delta) algorithm: it allows direct instead of batch updates. In [11] Sutton proposes the algorithms K1 and K2, two (linear) extensions to IDBD, and compares them with LMS and Kalman filtering. Koop [5] uses the IDBD algorithm for online adaptation and investigates general aspects of temporal coherence. Almeida [1] discusses another method of step-size adaptation and applies it to the minimization of nonlinear functions. Schraudolph [8] extends on the K1 algorithm and showed that it is superior to the approach described by Almeida [1].

Recently, Mahmood and Sutton [7] proposed with Autostep an extension to IDBD which has much less dependence on the meta-step-size parameter than IDBD. In the same year, Dabney and Barto [3] developed another adaptive step-size method for temporal difference learning, which is based on the estimation of upper and lower bounds. Again, both methods are proposed only for *linear* function approximation.

Schraudolph [9] and, more recently, Li [6] extended IDBD to the nonlinear case: Schraudolph's ELK1 performs an update with the instantaneous Hessian matrix of a suitable chosen loss function. The algorithm's complexity is $O(n^2)$ where n is the number of parameters to learn. This algorithm is superior to several others on the "four region" classification benchmark. However, this benchmark consists of a piecewise constant target function. Thus it does not exhibit steep and nonlinear slopes in the input-output-relationships which can be a major difficulty for adaptive learning, as we will show in this paper. – Li's KIMEL algorithm transforms the nonlinear input data with a kernel into a high-dimensional but linear feature space where linear IDBD is applied.

2 Methods

2.1 The Benchmark: A Nonlinear Nonstationary Task

In this work we consider a nonstationary task as a testbed as in [10], but with an additional nonlinearity: $n = 20$ real-valued inputs x_1, \dots, x_n are independently drawn from the standard normal distribution. The concept to learn is the weighted sum of the first 5 inputs, which is sent through a nonlinear function with slope σ_{nst}

$$y^* = \tanh \left(\sigma_{nst} \sum_{i=1}^5 s_i x_i \right) \quad (1)$$

where all the s_i are either $+1$ or -1 . To make this task nonstationary, one of the five s_i is selected randomly and switched in sign every 20 examples. Thus, the model has to learn that only the first five inputs are relevant and all other 15 inputs are irrelevant. At the same time the weights of the relevant inputs have to be able to change quickly in order to follow the drifting target.

2.2 Nonlinear Least Mean Squares (NLMS)

As a baseline learning algorithm we use a Nonlinear Least-Mean-Square (NLMS) model with constant learning rate α and error signal $\delta(t) = y^* - y(t)$:

$$y(t) = \tanh(N(t)) \quad \text{with} \quad N(t) = \sum_{i=1}^n w_i(t)x_i \quad (2)$$

$$\begin{aligned} w_i(t+1) &= w_i(t) + \alpha\delta(t) \frac{\partial y}{\partial w_i} \\ &= w_i(t) + \alpha\delta(t)(1 - y^2(t))x_i \end{aligned} \quad (3)$$

2.3 Incremental Delta Bar Delta (IDBD)

Sutton's IDBD algorithm [10] introduces for a *linear* unit $y(t) = \sum_i w_i x_i$ individual learning rates $\alpha_i = e^{\beta_i}$ for every weight w_i .

Algorithm 1. IDBD in pseudo code

```

1: Initialize:  $h_i = 0$ ,  $\beta_i = \beta_{init} \forall i$  and set  $\theta$ , the meta-learning rate.
2: for ( each new example  $(x_1, \dots, x_n, y^*)$  ) do
3:    $y = \sum_{i=1}^n w_i x_i$ 
4:    $\delta = y^* - y$ 
5:   for (every weight index  $i$  ) do
6:     Set  $\beta_i \leftarrow \beta_i + \theta x_i \delta h_i$ 
7:     Set  $\alpha_i \leftarrow e^{\beta_i}$ 
8:     Set  $w_i \leftarrow w_i + \alpha_i x_i \delta$ 
9:     Set  $h_i \leftarrow h_i [1 - \alpha_i x_i^2]^+ + \alpha_i x_i \delta$            with  $[d]^+ = d$  for  $d > 0$ ,  $=0$  else
10:  end for
11: end for

```

The main idea behind this algorithm is simple: The memory term h_i is a decaying trace of past weight changes. The increment in β_i is proportional to the product of the current weight change $x_i \delta$ and past weight changes h_i . Accumulated increments correspond to the *correlation* between current and recent weight changes [10]. In case of positive correlation the learning rate can be larger, while negative correlation indicates overshooting weight increments where the learning rate should be reduced.

2.4 Generalizing IDBD to Nonlinear Output Units

A simple approach to generalize IDBD to the nonlinear case would be to substitute the linear equation in Step 3 of the IDBD algorithm with the nonlinear Eq. (2). Then the difference is mainly the 'outer' derivation of the nonlinearity with respect to the net input $N(t)$ in Eq. (2). If we choose $\tanh(\cdot)$ as nonlinearity, this derivative yields the term $(1 - y^2)$ in several places. The explicit derivation will be shown later in Appendix A.

However, there is a severe problem with this simple approach: If the task exhibits a steep slope σ_{nst} in the nonlinear activation function, the adaptation of learning rates can quickly lead to a fully saturated system which does not learn the required concept. This is because large values of σ_{nst} lead to big error signals δ , and consequently to large learning rate changes and large weight changes. The output is driven into saturation sooner or later (near $+1$ or -1). With $1 - y^2 \approx 0$ the gradient information becomes unreliable. As a consequence, the mean squared error (MSE) will be big or even the whole system becomes unstable.

2.5 Controlling the Activation

To keep the average activation sufficiently small, we add an accumulator with

$$k_{acc}(t+1) = (1 - \gamma)k_{acc}(t) + \gamma [y(t)]^2 \quad \text{and} \quad k_{acc}(0) = 0, \quad (4)$$

where $\gamma = 0.001$ is a sufficiently small constant. It is easy to show that the corresponding initial value problem has the solution

$$k_{acc}(t) = \int_0^t y^2(\tau) \gamma e^{\gamma(\tau-t)} d\tau \quad (5)$$

For $t \gg 1/\gamma$ the function $f(\tau) = \gamma e^{\gamma(\tau-t)}$ plays the role of a density function, since $\int_0^t f(\tau) d\tau \approx 1$. Thus the accumulator $k_{acc}(t)$ is a **memory trace** of the square of recent activations. If for example the output is constant, $y(t) = y_0$, then $k_{acc}(t)$ will show an exponential decay towards y_0^2 . The smaller the parameter γ , the more long-term averaging the memory trace $k_{acc}(t)$ will be. The idea is now to add to the normal nonlinear weight update in Eq. (3) a new weight decay term proportional to $k_{acc}(t)$ with strength parameter ω_k

$$w_i(t+1) = w_i(t) + \alpha_i(t)\delta(t)(1 - y^2(t))x_i(t) - \omega_k k_{acc}(t)w_i(t)x_i^2(t) \quad (6)$$

The purpose of the weight decay term is as follows: If the average recent activation is high in absolute value (i.e. the activations are close to saturation), then all weights will be decaying to move the output out of the saturated zone. If on the other hand the activation is close to zero, then nearly no weight decay will take place. The special setting $\omega_k = 0$ allows to recover the 'old' situation without weight decay.

We summarize our new n-IDBD method in Algorithm 2. The main difference to (linear) IDBD is the term $Y = 1 - y^2$ in several places and the weight decay

Algorithm 2. n-IDBD: nonlinear IDBD in pseudo code

- 1: Initialize: $h_i = 0, \beta_i = \beta_{init} \forall i, \gamma = 0.001$, set the meta-learning rate θ , and set the weight decay parameter ω_k .
 - 2: **for** (each new example (x_1, \dots, x_n, y^*)) **do**
 - 3: Calculate y according to Eq. (2)
 - 4: Set $\delta = y^* - y, Y = 1 - y^2$ and $Z = Y + 2y\delta$
 - 5: Update accumulator $k_{acc} \leftarrow (1 - \gamma)k_{acc} + \gamma y^2$ according to Eq. (4)
 - 6: **for** (every weight index i) **do**
 - 7: Set $\beta_i \leftarrow \beta_i + \theta Y x_i h_i \delta$
 - 8: Set $\alpha_i \leftarrow e^{\beta_i}$
 - 9: Set $w_i \leftarrow w_i + \alpha_i Y x_i \delta - \omega_k k_{acc} w_i x_i^2$
 - 10: Set $h_i \leftarrow h_i [1 - (\alpha_i Y Z + \omega_k k_{acc}) x_i^2]^+ + \alpha_i Y x_i \delta$
 - 11: **end for**
 - 12: **end for**
-

term with $\omega_k k_{acc}$ in Steps 9 and 10 of the algorithm. It is a necessary prerequisite to achieve stable and fast learning. The precise form of the equations in Steps 7, 9, and 10 is derived in Appendix A.

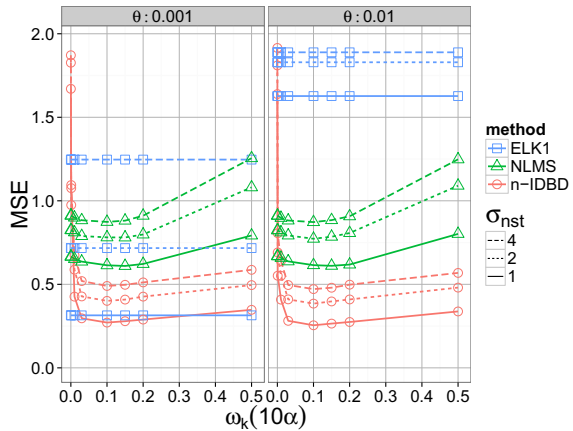


Fig. 1. Dependence on the weight decay parameter ω_K : n-IDBD has a broad minimum near $\omega_K = 0.1$. Shown is the asymptotic MSE at $t = 400\,000$. For NLMS the x-axis shows instead of ω_K the ten-fold learning rate 10α (i.e. we vary $\alpha \in [0, 0.05]$). For ELK1 there is no parameter ω_K .

3 Results

3.1 Does Weight Decay Help?

The first experiment answers the question whether n-IDBD performs better than ordinary NLMS (or LMS) on the benchmark task and what influence the weight decay has. In Fig. 1 we vary the weight decay parameter ω_k between 0 (no weight decay) and 0.5 (strong weight decay) and find a broad minimum near $\omega_k = 0.1$. The mean squared error (MSE) is taken at $t = 400\,000$ to get past any transient

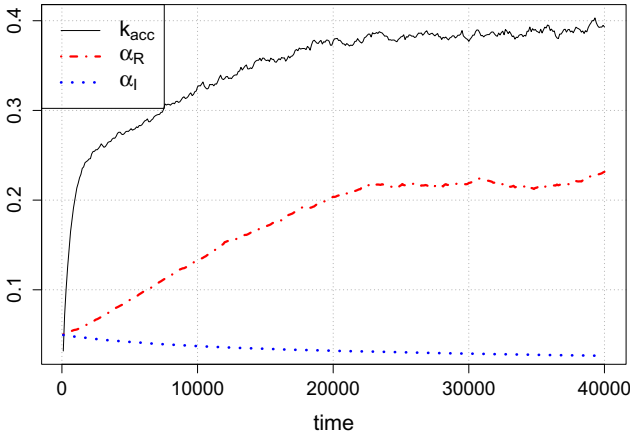


Fig. 2. Development of relevant / irrelevant learning rates (α_R, α_I) and accumulator k_{acc} . After 40 000 time steps n-IDBD adapts to the ratio $\alpha_R/\alpha_I \approx 9$.

phases. It is measured as the average of the squared error δ^2 between time steps 300 000 and 400 000.

Without weight decay the MSE rises sharply to values above 1.5. A closer inspection of the model shows that it is in this case fully saturated ($k_{acc} = 1$) with arbitrary large weights and large learning rates. It does not learn anything, it only jumps erratically between $+1$ and -1 . The learning rates surpass sensible bounds (e.g., $\alpha_i > 100$). A similar behavior is observed for ELK1 [8] which does not have any weight decay and exhibits in most cases large MSEs.

With weight decay the situation changes completely for a broad range of ω_k : n-IDBD has a low error everywhere, both weights and learning rates stabilize at roughly constant and sensible values. The overall activation of the unit stabilizes at a plateau $k_{acc} < 1$.¹ The MSE of n-IDBD is consistently lower than that of NLMS. This holds for all possible learning rates α of NLMS.

Fig. 2 shows the development of learning rates in one example of n-IDBD. Already after 10 000 time steps the algorithm differentiates well between relevant learning rates ($\alpha_R = \frac{1}{5} \sum_{i=1}^5 \alpha_i$) and irrelevant learning rates ($\alpha_I = \frac{1}{15} \sum_{i=6}^{20} \alpha_i$). After approximately 22 000 time steps both the learning rates and the activation k_{acc} stabilize at constant plateaus.

Fig. 3 compares the situation with and without weight decay again, but with a focus on the longer time scale. The model with weight decay is consistently better (has a lower MSE) than the one without. Whenever MSE becomes larger than 1.2, a closer inspection of the model shows that it is fully saturated ($k_{acc} = 1.0$) and the weights are unrealistically large. Even for very gentle slopes σ_{nst} , where the model without weight decay does not saturate (not shown in the figure), we find that weight decay is helpful to reduce the overall MSE.

¹ The precise value of $k_{acc} \in [0.2, 0.7]$ depends on the other algorithm parameters.

Fig. 3. Comparison of MSE without weight decay (circles, $\omega_K = 0$) to MSE with weight decay (triangles, $\omega_K = 0.1$). Shown is the MSE averaged over the last 100 000 time steps. The results with weight decay are always better.

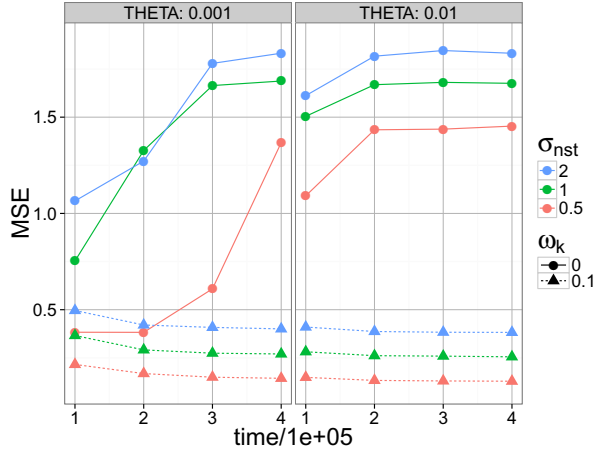
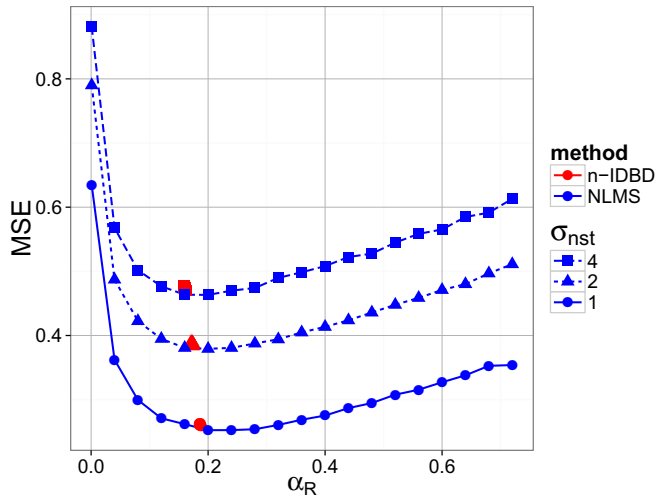


Fig. 4. Average error as a function of α_R , the learning rate for the relevant inputs. Irrelevant inputs had their weights clamped to zero in NLMS. The singular red point is the corresponding result from n-IDBD with $\theta = 0.01$.



3.2 Does Nonlinear IDBD Find the Optimal α_i ?

We have seen in the first experiment that n-IDBD automatically found large learning rates for the relevant weights and small learning rates for the irrelevant weights. Similar to [10] we want to test in a second experiment whether the learning rates found by n-IDBD are optimal. Therefore we build an 'ideal' NLMS for the task, where the irrelevant weights are already clamped to zero and the relevant weights get a predefined learning rate α_R . The 'ideal' NLMS has the same sigmoid and the same weight decay as n-IDBD. We get MSE-curves as shown in Fig. 4. The MSE is shown after 400 000 time steps, averaged between time steps 300 000 to 400 000. The red point for n-IDBD shows the average α_R (relevant weights) and the corresponding MSE. It is right at the minimum of each

σ_{nst}	Algorithm			
	LMS	NLMS	IDBD (linear)	n-IDBD (nonlinear)
1.0	0.54	0.61	0.34	0.25
2.0	0.70	0.78	0.47	0.38
4.0	0.79	0.87	0.56	0.47

Table 1. Comparison of the best MSE for all algorithms tested on the nonstationary task with varying slopes σ_{nst} . ‘Best’ means that each algorithm has its free parameters tuned to the best possible value. Parameters are $\theta = 0.01$, $\omega_k = 0.1$ for IDBD, $\alpha = 0.2$ for LMS, and $\alpha = 0.1$ for NLMS.

ideal curve. This shows that there is no other setting of learning rate parameters which will perform better. (The MSE is slightly higher for n-IDBD than for NLMS, because n-IDBD has the irrelevant weights not clamped precisely to zero, they fluctuate at a small level.)

We finally compare in Tab. 1 the best MSE for all algorithms. It is remarkable that linear IDBD is better on the task than nonlinear NLMS. But n-IDBD is clearly better than all other tested algorithms for all sigmoidal slopes σ_{nst} .

4 Conclusion

We have extended the adaptive, linear IDBD to the nonlinear case. It was shown that a simple extension would lead to an instable nonlinear system due to saturation effects. Similarly, the well-known ELK1 method showed diverging weights for most parameter settings as well. We proposed an additional self-regulative mechanism to control the average activation. This makes the adaptive system stable again. As in the linear case [10], the adaptive system finds the best possible learning rate on the benchmark task. The n-IDBD algorithm exhibits a smaller MSE on the benchmark task than either LMS, NLMS or linear IDBD. In an upcoming paper [2] we will show that n-IDBD can be applied to a game-learning task (Connect-Four) with more than half a million of weights as well.

A Appendix: Derivation of n-IDBD

Similar to [10], the equations of n-IDBD can be derived from a few simple principles. We start with two loss functions

$$L_1(t) = \frac{1}{2}\delta^2(t) \quad \text{and} \quad L_2(t) = \frac{1}{2} \sum_i w_i^2(t)x_i^2(t). \quad (7)$$

Both $L_1(t)$ and $L_2(t)$ should be minimized by the learning algorithm. The first term rewards small errors and the second term regularizes the complexity of the network: Weights with active inputs ($x_i^2 > 0$) should be as small as possible in their square sum.² In each learning step a weight change will be made in the steepest-descent direction for L_1 and for L_2 :

² It is also possible to use a simpler $L_2 = \frac{1}{2}\sum_i w_i^2(t)$ without the term $x_i^2(t)$. Then every weight decays in each time step. This leads to the same qualitative results in the benchmark task of Sec. 2.1, but might lead to different results in larger systems with sparse input activations.

$$\begin{aligned}
 w_i(t+1) &= w_i(t) - \alpha \frac{\partial L_1(t)}{\partial w_i(t)} - \Omega \frac{\partial L_2(t)}{\partial w_i(t)} \\
 &= w_i(t) - \alpha \delta(t) \frac{\partial \delta(t)}{\partial w_i(t)} - \Omega w_i(t) x_i^2(t) \\
 &= w_i(t) + \alpha \delta(t) \frac{\partial y(t)}{\partial N(t)} \frac{\partial N(t)}{\partial w_i(t)} - \Omega w_i(t) x_i^2(t) \\
 &= w_i(t) + \alpha \delta(t) (1 - y^2(t)) x_i(t) - \Omega w_i(t) x_i^2(t) \tag{8}
 \end{aligned}$$

with constants α and Ω . If we identify the constant α with the slowly varying learning rate α_i and the constant Ω with $\omega_k k_{acc}(t)$ (which is justified, because $k_{acc}(t)$ approaches – after a transient phase – a roughly constant value), then Eq. (8) reproduces the weight update rule Eq. (6) for n-IBDB.

The β -update rule is governed by the minimization of L_1

$$\begin{aligned}
 \beta_i(t+1) &= \beta_i(t) - \theta \frac{\partial L_1(t)}{\partial \beta_i(t)} \\
 &= \beta_i(t) - \theta \delta(t) \frac{\partial \delta(t)}{\partial \beta_i(t)} \\
 &= \beta_i(t) + \theta \delta(t) \frac{\partial y(t)}{\partial N(t)} \frac{\partial N(t)}{\partial w_i(t)} \frac{\partial w_i(t)}{\partial \beta_i(t)} \\
 &= \beta_i(t) + \theta \delta(t) (1 - y^2(t)) x_i(t) h_i(t) \tag{9}
 \end{aligned}$$

where we have defined $h_i(t) \equiv \frac{\partial w_i(t)}{\partial \beta_i(t)}$ as in [10]. We abbreviate $Y(t) \equiv (1 - y^2(t))$ and derive the h -update rule:

$$\begin{aligned}
 h_i(t+1) &= \frac{\partial}{\partial \beta_i} w_i(t+1) \\
 &= \frac{\partial}{\partial \beta_i} (w_i(t) + e^{\beta_i} \delta(t) Y(t) x_i(t) - \Omega w_i(t) x_i^2(t)) \\
 &= h_i(t) + \left[e^{\beta_i} \delta(t) Y(t) + e^{\beta_i} \frac{\partial \delta(t)}{\partial \beta_i} Y(t) + e^{\beta_i} \delta(t) \frac{\partial Y(t)}{\partial \beta_i} \right] x_i(t) - \Omega h_i(t) x_i^2(t) \\
 &= h_i(t) + \alpha_i \left[\delta(t) Y(t) + \frac{\partial \delta(t)}{\partial \beta_i} Y(t) + \delta(t) \frac{\partial Y(t)}{\partial \beta_i} \right] x_i(t) - \Omega h_i(t) x_i^2(t) \tag{10}
 \end{aligned}$$

The terms in square brackets come from the threefold product rule when taking the partial derivative of $e^{\beta_i} \delta(t) Y(t)$ with respect to β_i . We know from Eq. (9) that

$$\frac{\partial \delta(t)}{\partial \beta_i} = -Y(t) x_i(t) h_i(t)$$

Similarly we obtain

$$\frac{\partial Y(t)}{\partial \beta_i} = \frac{\partial (1 - y^2(t))}{\partial \beta_i} = -2y(t) \frac{\partial y(t)}{\partial \beta_i} = -2y(t) Y(t) x_i(t) h_i(t)$$

If we put everything together and collect terms in Eq. (10), it is straightforward to derive

$$h_i(t+1) = h_i(t) [1 - (\alpha_i Y(t) Z(t) + \Omega) x_i^2(t)] + \alpha_i \delta(t) Y(t) x_i(t) \quad (11)$$

with $Z(t) = Y(t) + 2y(t)\delta(t)$

which is, after adding a positive-bounding operation for the term in square brackets, the h -update rule of n-IDBD. Here the Ω -term ensures stability as well: Even close to saturation (when $y^2(t) \approx 1$, hence $Y(t) \approx 0$) the Ω -term guarantees the decay of h_i .

References

1. Almeida, L., Langlois, T., Amaral, J.D.: On-line step size adaptation. Technical Report RT07/97, INESC. 9 Rua Alves Redol, Lisboa, Portugal (1997)
2. Bagheri, S., Thill, M., Koch, P., Konen, W.: Online adaptable learning rates for the game Connect-4. Submitted to IEEE Trans. on Computational Intelligence and AI in Games (2014)
3. Dabney, W., Barto, A.G.: Adaptive step-size for online temporal difference learning. In: 26th AAAI Conference on Artificial Intelligence (2012)
4. Jacobs, R.A.: Increased rates of convergence through learning rate adaptation. *Neural Networks* 1(4), 295–307 (1988)
5. Koop, A.: Investigating experience: Temporal coherence and empirical knowledge representation. Master's thesis, University of Alberta, Canada (2008)
6. Li, C., Ye, Y., Miao, Q., Shen, H.-L.: Kimel: A kernel incremental metalearning algorithm. *Signal Processing* 93(6), 1586–1596 (2013); Special issue on Machine Learning in Intelligent Image Processing
7. Mahmood, A.R., Sutton, R.S., Degris, T., Pilarski, P.M.: Tuning-free step-size adaptation. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2121–2124. IEEE (2012)
8. Schraudolph, N.N.: Online local gain adaptation for multi-layer perceptrons. Technical Report IDSIA-09-98, Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale (IDSIA), Switzerland (1998)
9. Schraudolph, N.N.: Online learning with adaptive local step sizes. In: *Neural Nets WIRN Vietri 1999*, pp. 151–156. Springer (1999)
10. Sutton, R.S.: Adapting bias by gradient descent: An incremental version of delta-bar-delta. In: Swartout, W.R. (ed.) 10th AAAI Conference on Artificial Intelligence, pp. 171–176 (1992)
11. Sutton, R.S.: Gain adaptation beats least squares. In: 7th Yale Workshop on Adaptive and Learning Systems, pp. 161–166 (1992)

On the Effectiveness of Sampling for Evolutionary Optimization in Noisy Environments^{*}

Chao Qian¹, Yang Yu¹, Yaochu Jin², and Zhi-Hua Zhou¹

¹ National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China

² Department of Computing, University of Surrey, Guildford, Surrey, GU2 7XH, UK
{qianc, yuy, zhouzh}@lamda.nju.edu.cn, yaochu.jin@surrey.ac.uk

Abstract Sampling has been often employed by evolutionary algorithms to cope with noise when solving noisy real-world optimization problems. It can improve the estimation accuracy by averaging over a number of samples, while also increasing the computation cost. Many studies focused on designing efficient sampling methods, and conflicting empirical results have been reported. In this paper, we investigate the effectiveness of sampling in terms of rigorous running time, and find that sampling can be ineffective. We provide a general sufficient condition under which sampling is useless (i.e., sampling increases the running time for finding an optimal solution), and apply it to analyzing the running time performance of (1+1)-EA for optimizing OneMax and Trap problems in the presence of additive Gaussian noise. Our theoretical analysis indicates that sampling in the above examples is not helpful, which is further confirmed by empirical simulation results.

1 Introduction

Evolutionary algorithms (EAs) [4] inspired from natural phenomena are often applied to solve real-world optimization problems, where the fitness (i.e., objective) evaluation of a solution is usually noisy. For example, in airplane design, the fitness of every prototype is evaluated by a stochastic computer simulation, and thus is a random variable whose value can be different from the exact fitness. Handling noise in fitness evaluations is important in that a poor solution can appear to be good due to the noise, which can mislead the search direction, resulting in an inefficient optimization. Many studies thus have focused on dealing with noise in evolutionary optimization [2,6,18].

One simple and direct way to reduce the effect of noise is sampling, which samples the fitness of one solution several times and then uses the average to estimate the true fitness. An n -time random sampling can reduce the standard deviation by a factor of \sqrt{n} , and thus makes the fitness estimation closer to the true value, while also increasing the computation cost n times. Much effort has been devoted to designing smarter sampling approaches, which dynamically decide the sample size for each solution so that the sampling cost is reduced as much as possible.

^{*} This research was supported by the National Science Foundation of China (61375061, 61333014) and the Jiangsu Science Foundation (BK2012303).

Aizawa and Wah [1] suggested two adaptive sampling methods: increasing the sample size with the generation number and allocating larger sample size for solutions with larger estimated variance. Stagge [24] used a larger sample size for better solutions. Several sequential sampling approaches [7,8,10] were later proposed for tournament selection, which first estimate the fitness of two solutions by a small number of samples, and then sequentially increase samples until the difference can be significantly discriminated. Adaptive sampling was then incorporated into diverse metaheuristic algorithms (e.g., immune algorithm [27], particle swarm optimization [5] and compact differential evolution [17]) to efficiently cope with noise. It has also been employed by evolutionary algorithms for noisy multi-objective optimization [20,23,25]. Based on the assumption that the fitness landscape is locally smooth, an alternative approach to approximately increase the estimation accuracy without increasing the sampling cost was proposed [9,22], which estimates the fitness of a solution by averaging the fitness of previously evaluated neighbors.

Sampling has been shown to be able to improve the local performance of EAs (e.g., increase the probability of selecting the true better solution in tournament selection [7]). A practical performance measure of an algorithm is how much time it needs to find a desired solution. On this measure, conflicting conclusions about sampling have been empirically reported. For example, in [1], it was shown that sampling can speed up a standard genetic algorithm on two test functions; while in [10], sampling led to a larger computation time for a simple generational genetic algorithm on the OneMax function.

In this paper, we investigate the effectiveness of sampling via rigorous running time analysis, which measures how soon an algorithm can solve a problem (i.e., the number of fitness evaluations until finding an optimal solution) and has been a leading theoretical aspect for randomized search heuristics [3,19]. We provide a sufficient condition under which sampling is useless (i.e., sampling increases the running time). Applying it to analyze (1+1)-EA solving the Noisy OneMax and the Noisy Trap problems with the additive Gaussian noise, we disclose that the sampling is ineffective in the two cases for different reasons. The derived theoretical results are also empirically verified. The results may help understand the effect of noise and design better strategies for handling noisy fitness functions.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. Section 3 presents the main theorem, which is then used in case studies in Section 4. Section 5 concludes the paper and discusses future work.

2 Preliminaries

2.1 Sampling and Optimization in the Presence of Noise

An optimization problem can be generally represented as $\arg \max_{x \in \mathcal{X}} f(x)$, where \mathcal{X} is the feasible solution space and the objective f is also called fitness in the context of evolutionary computation. In real-world optimization tasks, the fitness evaluation for a solution is usually disturbed by noise due to a wide range of uncertainties (e.g., randomized simulations), and consequently we can not obtain the exact fitness value but only a noisy one. A commonly studied noise model is additive noise as presented in Definition 1, which will also be adopted in this paper.

Definition 1 (Additive Noise). Given a distribution \mathcal{N} , let $f^{\mathcal{N}}(x)$ and $f(x)$ denote the noisy and true fitness of a solution x respectively, then

$$f^{\mathcal{N}}(x) = f(x) + \delta,$$

where δ is randomly drawn from \mathcal{N} , denoted by $\delta \sim \mathcal{N}$.

In evolutionary optimization, sampling as described in Definition 2 has often been used to reduce the effect of noise. It approximates the true fitness $f(x)$ by the average of a number of random samples.

Definition 2 (Sampling). Sampling of size k outputs the fitness of a solution as

$$f_k^{\mathcal{N}}(x) = \frac{1}{k} \sum_{i=1}^k (f(x) + \delta_i), \text{ where } \delta_i \sim \mathcal{N}.$$

For additive Gaussian noise (i.e., $\mathcal{N} = N(\theta, \sigma^2)$), $f_k^{\mathcal{N}}(x)$ actually can be represented by $f(x) + \delta$ with $\delta \sim N(\theta, \sigma^2/k)$, that is, sampling of size k reduces the variance of noise by a factor of k and thus estimates the fitness more accurately. However, the computation time is also increased by k times. Many studies thus focused on designing efficient sampling methods [1,8,24], while the effectiveness of sampling, in particular a theoretical understanding of sampling, remains unclear.

2.2 Evolutionary Algorithms by Markov Chain Analysis

Evolutionary algorithms [4] are a kind of randomized metaheuristic optimization algorithms. Starting from an initial set of solutions (called a population), EAs try to iteratively improve the population by a cycle of three stages: reproducing new solutions from the current population, evaluating the newly generated solutions, and updating the population by removing bad solutions. The (1+1)-EA, as shown in Algorithm 1, is a simple EA for maximizing pseudo-Boolean problems over $\{0, 1\}^n$, which reflects the common structure of EAs. It maintains only one solution, and repeatedly tries to improve the current solution by using bit-wise mutation (i.e., step 3) and selection (i.e., steps 4-5). It has been widely used for the running time analysis of EAs, e.g., in [16,26]. For (1+1)-EA with sampling in noisy environments, step 4 changes to be “if $f_k^{\mathcal{N}}(x') \geq f_k^{\mathcal{N}}(x)$ ”.

Algorithm 1 ((1+1)-EA). Given pseudo-Boolean function f with solution length n , it consists of the following steps:

1. $x :=$ randomly selected from $\{0, 1\}^n$.
2. Repeat until the termination condition is met
3. $x' :=$ flip each bit of x independently with probability p .
4. if $f(x') \geq f(x)$
5. $x := x'$.

where $p \in (0, 0.5)$ is the mutation probability.

The evolution process goes forward only based on the current population, thus, an EA can be modeled and analyzed as a Markov chain $\{\xi_t\}_{t=0}^{+\infty}$ (e.g., in [16,26]) by taking the EA's population space \mathcal{X} as the chain's state space, i.e. $\xi_t \in \mathcal{X}$. Let $\mathcal{X}^* \subset \mathcal{X}$ denote

the set of all optimal populations, which contains at least one optimal solution. The goal of the EA is to reach \mathcal{X}^* from an initial population. Thus, \mathcal{X}^* is the optimal state space of the corresponding Markov chain. In this paper, we assume that the Markov chain is homogeneous, since EAs often employ time-invariant operators.

Given a Markov chain $\{\xi_t\}_{t=0}^{+\infty}$ and $\xi_t = x$, we define its *first hitting time* (FHT) as a random variable τ such that $\tau = \min\{t | \xi_{t+t} \in \mathcal{X}^*, t \geq 0\}$. That is, τ is the number of steps needed to reach the optimal space for the first time starting from $\xi_t = x$. The mathematical expectation of τ , $\mathbb{E}[\tau | \xi_t = x] = \sum_{i=0}^{+\infty} iP(\tau = i)$, is called the *expected first hitting time* (EFHT). For the corresponding EA, the running time is usually defined as the number of fitness evaluations until an optimal solution is found for the first time, since the fitness evaluation is often the computational process with the highest cost [16,26]. Thus, the *expected running time* of the EA starting from ξ_0 is equal to $N_1 + N_2 \cdot \mathbb{E}[\tau | \xi_0]$, where N_1 and N_2 are the number of fitness evaluations for the initial population and each iteration, respectively. For example, for (1+1)-EA without noise, $N_1 = 1$ and $N_2 = 1$. Note that, for EAs under noise, we assume that the reevaluation strategy [13,14,18] is used, i.e., when accessing the fitness of a solution, it is always reevaluated. For example, for (1+1)-EA with sampling, both $f_k^N(x')$ and $f_k^N(x)$ will be calculated and recalculated in each iteration; thus, $N_1 = k$ and $N_2 = 2k$.

Lemma 1 characterizing the EFHT of a Markov chain by one-step transition and Lemma 2 showing the drift analysis tool will be used to analyze the EFHT of Markov chains in the paper. Drift analysis was first introduced to the running time analysis of EAs by He and Yao [16] and later many variants have been proposed (e.g., in [11,12]). To use it, a function $V(x)$ has to be constructed to measure the distance of a state x to the optimal state space \mathcal{X}^* . The distance function $V(x)$ satisfies that $V(x \in \mathcal{X}^*) = 0$ and $V(x \notin \mathcal{X}^*) > 0$. Then, by investigating the progress on the distance to \mathcal{X}^* in each step, i.e., $\mathbb{E}[V(\xi_t) - V(\xi_{t+1}) | \xi_t]$, an upper (lower) bound of the EFHT can be derived through dividing the initial distance by a lower (upper) bound of the progress.

Lemma 1. *Given a Markov chain $\{\xi_t\}_{t=0}^{+\infty}$, we have*

$$\begin{aligned} \forall x \in \mathcal{X}^* : \mathbb{E}[\tau | \xi_t = x] &= 0; \\ \forall x \notin \mathcal{X}^* : \mathbb{E}[\tau | \xi_t = x] &= 1 + \sum_{y \in \mathcal{X}} P(\xi_{t+1} = y | \xi_t = x) \mathbb{E}[\tau | \xi_{t+1} = y]. \end{aligned}$$

Lemma 2 (Drift Analysis [16]). *Given a Markov chain $\{\xi_t\}_{t=0}^{+\infty}$ and a distance function $V(x)$, if it satisfies that for any $t \geq 0$ and any ξ_t with $V(\xi_t) > 0$,*

$$0 < c_l \leq \mathbb{E}[V(\xi_t) - V(\xi_{t+1}) | \xi_t] \leq c_u,$$

then the EFHT satisfies that $V(\xi_0)/c_u \leq \mathbb{E}[\tau | \xi_0] \leq V(\xi_0)/c_l$.

3 Theorem on Sampling Effectiveness

In this section, for EAs solving noisy problems, we provide two situations where the running time increases with the sample size, i.e., sampling is useless. Let $\{\xi'_t\}_{t=0}^{+\infty}$ model the evolutionary process without noise, and let $\{\xi_t\}_{t=0}^{+\infty}$ model that using the sample

size k for fitness evaluation under noise. We always denote \mathcal{X} and \mathcal{X}^* as the state space and the optimal state space, respectively. For any $x, x' \in \mathcal{X}$, let $p(x, x')$ and $q_k(x, x')$ denote the probability of jumping from state x to x' in one step for $\{\xi'_t\}_{t=0}^{+\infty}$ and $\{\xi_t\}_{t=0}^{+\infty}$ respectively, i.e., $p(x, x') = P(\xi'_{t+1} = x' | \xi'_t = x)$ and $q_k(x, x') = P(\xi_{t+1} = x' | \xi_t = x)$. For clarity, we also represent the EFHT of $\{\xi'_t\}_{t=0}^{+\infty}$ and $\{\xi_t\}_{t=0}^{+\infty}$ by $\mathbb{E}[x]$ and $\mathbb{E}_k[x]$ respectively, i.e., $\mathbb{E}[x] = \mathbb{E}[\tau' | \xi'_0 = x]$ and $\mathbb{E}_k[x] = \mathbb{E}[\tau | \xi_0 = x]$. Let $\frac{dg(k)}{dk}$ denote the derivative of a function $g(k)$ with respect to k .

Theorem 1. For an EA \mathcal{A} optimizing a problem f under some kind of noise, if there exists a function $g(k)$ ($k \geq 1$) such that either one of the following two situations holds,

- (1) $\max_{x \notin \mathcal{X}^*} \left\{ \sum_{x': \mathbb{E}[x'] \neq \mathbb{E}[x]} (q_k(x, x') - p(x, x')) (\mathbb{E}[x] - \mathbb{E}[x']) \right\} \leq g(k) < 0,$
 and $1 + g(k) - k \frac{dg(k)}{dk} \geq 0;$
- (2) $\min_{x \notin \mathcal{X}^*} \left\{ \sum_{x': \mathbb{E}[x'] \neq \mathbb{E}[x]} (q_k(x, x') - p(x, x')) (\mathbb{E}[x] - \mathbb{E}[x']) \right\} \geq g(k) > 0,$ and $\frac{dg(k)}{dk} \leq 0,$

then for any $x \in \mathcal{X}$, $k \cdot \mathbb{E}_k[x] \leq (k + 1) \cdot \mathbb{E}_{k+1}[x]$, i.e., sampling is useless.

Before the proof, we first intuitively explain these two situations where sampling is useless. In situation (1), noise is harmful and using a larger sample size may reduce its negative effect (i.e., $\mathbb{E}_k(x)$ decreases with k), but the decrease rate of $\mathbb{E}_k(x)$ is smaller than the increase rate of the sample size k ; thus sampling is overall useless. In situation (2), noise is actually helpful and using a larger sample size reduces its positive effect, thus $\mathbb{E}_k(x)$ increases with k and sampling is of course useless.

Proof. We use Lemma 2 to prove a bound on $\mathbb{E}_k[x]$. We first construct a distance function $\forall x \in \mathcal{X}$, $V(x) = \mathbb{E}[x]$, which satisfies that $V(x \in \mathcal{X}^*) = 0$ and $V(x \notin \mathcal{X}^*) > 0$ by Lemma 1. Then, we investigate $\mathbb{E}[V(\xi_t) - V(\xi_{t+1}) | \xi_t = x]$ for any $x \notin \mathcal{X}^*$.

$$\begin{aligned} \mathbb{E}[V(\xi_t) - V(\xi_{t+1}) | \xi_t = x] &= V(x) - \mathbb{E}[V(\xi_{t+1}) | \xi_t = x] \\ &= 1 + \sum_{x' \in \mathcal{X}} p(x, x') \mathbb{E}[x'] - \sum_{x' \in \mathcal{X}} q_k(x, x') \mathbb{E}[x'] \quad (\text{by Lemma 1}) \\ &= 1 + \sum_{x': \mathbb{E}[x'] \neq \mathbb{E}[x]} (q_k(x, x') - p(x, x')) (\mathbb{E}[x] - \mathbb{E}[x']) =: 1 + g(x, k). \end{aligned}$$

If situation (1) holds, $\mathbb{E}[V(\xi_t) - V(\xi_{t+1}) | \xi_t = x] \leq 1 + g(k)$. By Lemma 2, we have $\mathbb{E}_k[x] \geq \mathbb{E}[x] / (1 + g(k))$, which shows that noise is harmful since $g(k) < 0$. The expected running time starting from x can be represented by $Mk + Nk \cdot \mathbb{E}_k[x]$, where M and N denote the number of solutions that need to be evaluated for the initial population and each iteration, respectively. The EFHT $\mathbb{E}_k[x]$ may decrease with k ; thus we need to compare its decrease rate with the increase rate of the sample size k . The condition $1 + g(k) - k \frac{dg(k)}{dk} \geq 0$ implies that $\frac{d(k/(1+g(k)))}{dk} \geq 0$, i.e., the decrease rate of $\mathbb{E}_k[x]$ is smaller than the increase rate of k . Thus, the expected running time starting from x increases with k , i.e., sampling is useless.

If situation (2) holds, $\mathbb{E}[V(\xi_t) - V(\xi_{t+1}) | \xi_t = x] \geq 1 + g(k)$. By Lemma 2, we have $\mathbb{E}_k[x] \leq \mathbb{E}[x] / (1 + g(k))$, which shows that noise is helpful since $g(k) > 0$. Due to that $\frac{dg(k)}{dk} \leq 0$ (i.e., $g(k)$ decreases with k), $\mathbb{E}_k[x]$ increases with k . Thus, the expected running time $Mk + Nk \cdot \mathbb{E}_k[x]$ obviously increases with k , i.e., sampling is useless. \square

4 Case Studies

In this section, we will apply the above theorem to analyze the effectiveness of sampling for EAs solving different pseudo-Boolean problems under additive Gaussian noise.

4.1 (1+1)-EA on Noisy OneMax

OneMax problem is to maximize the number of 1 bits of a solution $x \in \{0, 1\}^n$. It has become a benchmark for the running time analysis of EAs; particularly, the expected running time of (1+1)-EA with mutation probability $\frac{1}{n}$ is $\Theta(n \log n)$ [15]. For its noisy variant as in Definition 3, the fitness of a solution accessed in the optimization is a noisy one $f^{\mathcal{N}}(x)$ instead of the true fitness $f(x)$.

Definition 3 (Noisy OneMax Problem). *Given a distribution \mathcal{N} and $n \in \mathbb{N}^+$, defining*

$$f^{\mathcal{N}}(x) = f(x) + \delta = \sum_{i=1}^n x_i + \delta$$

where $x \in \{0, 1\}^n$ and δ is randomly drawn from \mathcal{N} , Noisy OneMax Problem of size n is to solve the problem: $\arg \max_{x \in \{0, 1\}^n} \mathbb{E}_{\delta \sim \mathcal{N}}[f^{\mathcal{N}}(x)]$.

Theorem 2. *For any $\sigma > 0$, sampling is useless for (1+1)-EA optimizing Noisy OneMax problem with Gaussian noise $\mathcal{N} = N(\theta, \sigma^2)$.*

Proof. We are to show that the situation (1) of Theorem 1 holds here. From Lemma 1 in [21], we know that the EFHT $\mathbb{E}[x]$ of (1+1)-EA on OneMax without noise depends on the number of 0 bits $|x|_0$ and increases with it. Let $\text{mut}(x, x')$ denote the probability of mutating from x to x' by step 3 of Algorithm 1. Note that, by sampling of size k , the Gaussian noise reduces to be $N(\theta, \sigma^2/k)$, i.e., $f_k^{\mathcal{N}}(x) = f(x) + \delta$ with $\delta \sim N(\theta, \sigma^2/k)$.

For any x with $|x|_0 = i \geq 1$ and x' with $|x'|_0 = j$, if $j < i$, $p(x, x') = \text{mut}(x, x')$ since x' has less 0 bits and is better than x ; if $j > i$, $p(x, x') = 0$ since x' has more 0 bits and is worse than x . We also have $q_k(x, x') = \text{mut}(x, x') \cdot \text{Prob}(f(x') + \delta_1 \geq f(x) + \delta_2)$, where $\delta_1, \delta_2 \sim N(\theta, \sigma^2/k)$. Note that $\delta_1 - \delta_2 \sim N(0, 2\sigma^2/k)$. Thus, $q_k(x, x') = \text{mut}(x, x') \cdot \text{Prob}(\delta \geq j - i)$, where $\delta \sim N(0, 2\sigma^2/k)$. Then, we have

$$\begin{aligned} g(x, k) &= \sum_{x': \mathbb{E}[x'] \neq \mathbb{E}[x]} (q_k(x, x') - p(x, x')) (\mathbb{E}[x] - \mathbb{E}[x']) \\ &= - \sum_{|x'|_0 = j < i} \text{mut}(x, x') \cdot \text{Prob}(\delta > i - j) \cdot (\mathbb{E}[x] - \mathbb{E}[x']) \\ &\quad + \sum_{|x'|_0 = j > i} \text{mut}(x, x') \cdot \text{Prob}(\delta \geq j - i) \cdot (\mathbb{E}[x] - \mathbb{E}[x']) \\ &\leq - \text{Prob}(\delta > 1) \cdot \left(\sum_{|x'|_0 = i-1} \text{mut}(x, x') (\mathbb{E}[x] - \mathbb{E}[x']) \right). \end{aligned}$$

Let $c = \min_{x \notin \mathcal{X}^*} \sum_{|x'|_0 = i-1} \text{mut}(x, x') (\mathbb{E}[x] - \mathbb{E}[x'])$. Let $\delta' \sim N(0, 1)$. Then, $\text{Prob}(\delta > 1) = \text{Prob}(\delta' > \frac{\sqrt{k}}{\sqrt{2}\sigma})$. By $\text{Prob}(\delta' > m) \geq \frac{m}{2\sqrt{2\pi}} e^{-m^2/2}$ for $0 < m \leq 1$, we can get $\text{Prob}(\delta > 1) \geq \frac{\sqrt{k}}{4\sigma\sqrt{\pi}} \cdot e^{-k/4\sigma^2}$ when $k \leq 2\sigma^2$. Thus, let $g(k) = -\frac{c\sqrt{k}}{4\sigma\sqrt{\pi}} \cdot e^{-k/4\sigma^2}$ which satisfies that $\max_{x \notin \mathcal{X}^*} g(x, k) \leq g(k) < 0$. Then,

$$1 + g(k) - k \frac{dg(k)}{dk} = 1 - \frac{c\sqrt{k}}{8\sigma\sqrt{\pi}} \cdot e^{-k/4\sigma^2} - \frac{ck\sqrt{k}}{16\sigma^3\sqrt{\pi}} \cdot e^{-k/4\sigma^2} \geq 1 - \frac{c}{2\sqrt{2\pi}}.$$

When $k \geq 2\sigma^2$, $Prob(\delta > 1) \approx \frac{\sigma}{\sqrt{k\pi}} \cdot e^{-k/4\sigma^2}$, since $Prob(\delta' > m) \approx \frac{1}{m} \cdot \frac{1}{\sqrt{2\pi}} e^{-m^2/2}$ for $m \geq 1$. Thus, let $g(k) = -\frac{c\sigma}{\sqrt{k\pi}} \cdot e^{-k/4\sigma^2}$. Then,

$$1 + g(k) - k \frac{dg(k)}{dk} = 1 - \frac{3c\sigma}{2\sqrt{k\pi}} \cdot e^{-k/4\sigma^2} - \frac{c\sqrt{k}}{4\sigma\sqrt{\pi}} \cdot e^{-k/4\sigma^2} \geq 1 - \frac{\sqrt{2}c}{\sqrt{\pi}} e^{-1/2},$$

where the inequality is since xe^{-x^2} reaches the maximum when $x = \frac{\sqrt{2}}{2}$. Then, we are to show that $c \leq 1$. By Lemma 1, for any x with $|x|_0 = i \geq 1$, we have

$$\begin{aligned} \mathbb{E}[x] &= 1 + \sum_{j=0}^{i-1} \sum_{|x'|_0=j} mut(x, x') \mathbb{E}[x'] + (1 - \sum_{j=0}^{i-1} \sum_{|x'|_0=j} mut(x, x')) \mathbb{E}[x] \\ &\leq 1 + (\sum_{j=0}^{i-1} \sum_{|x'|_0=j} mut(x, x')) \mathbb{E}[x' \mid |x'|_0 = i - 1] + (1 - \sum_{j=0}^{i-1} \sum_{|x'|_0=j} mut(x, x')) \mathbb{E}[x]. \end{aligned}$$

Thus, $\mathbb{E}[x] - \mathbb{E}[x' \mid |x'|_0 = i - 1] \leq 1 / \sum_{j=0}^{i-1} \sum_{|x'|_0=j} mut(x, x')$. Then,

$$\sum_{|x'|_0=i-1} mut(x, x') (\mathbb{E}[x] - \mathbb{E}[x']) \leq \sum_{|x'|_0=i-1} mut(x, x') / \sum_{j=0}^{i-1} \sum_{|x'|_0=j} mut(x, x') \leq 1,$$

which implies that $c \leq 1$. Thus, $1 + g(k) - k \frac{dg(k)}{dk} \geq 0$. □

4.2 (1+1)-EA on Noisy Trap

Trap problem is another commonly used problem in the theoretical analysis of EAs. It is to maximize the number of 0 bits of a solution except the global optimum 11...1; the expected running time of (1+1)-EA with mutation probability $\frac{1}{n}$ is $\Theta(n^n)$ [15].

Definition 4 (Noisy Trap Problem). Given a distribution \mathcal{N} and $n \in \mathbb{N}^+$, defining

$$f^{\mathcal{N}}(x) = f(x) + \delta = C \prod_{i=1}^n x_i - \sum_{i=1}^n x_i + \delta$$

where $x \in \{0, 1\}^n$, $C > n$ and δ is randomly drawn from \mathcal{N} , Noisy Trap Problem of size n is to solve the problem: $\arg \max_{x \in \{0, 1\}^n} \mathbb{E}_{\delta \sim \mathcal{N}} [f^{\mathcal{N}}(x)]$.

Theorem 3. For any $\sigma > 0$, sampling is useless for (1+1)-EA optimizing Noisy Trap problem with Gaussian noise $\mathcal{N} = N(\theta, \sigma^2)$ and $C = +\infty$.

Proof. We are to show that the situation (2) of Theorem 1 holds here. From Lemma 2 in [21], we know that the EFHT $\mathbb{E}[x]$ of (1+1)-EA on Trap without noise depends on $|x|_0$ and increases with it.

For any x with $|x|_0 = i \geq 1$ and x' with $|x'|_0 = j$, $p(x, x') = 0$ if $0 < j < i$, and $p(x, x') = mut(x, x')$ if $j = 0$ or $j > i$; $q_k(x, x') = mut(x, x') \cdot Prob(\delta \geq i - j)$ if $j > 0$ and $q_k(x, x') = mut(x, x') \cdot Prob(\delta \geq i - C)$ if $j = 0$, where $\delta \sim N(0, 2\sigma^2/k)$. Note that $C = +\infty$, thus $q_k(x, x') = mut(x, x')$ if $j = 0$. Then, we have

$$\begin{aligned}
 g(x, k) &= \sum_{x': \mathbb{E}[x'] \neq \mathbb{E}[x]} (q_k(x, x') - p(x, x'))(\mathbb{E}[x] - \mathbb{E}[x']) \\
 &= \sum_{0 < |x'|_0 = j < i} mut(x, x') \cdot Prob(\delta \geq i - j) \cdot (\mathbb{E}[x] - \mathbb{E}[x']) \\
 &\quad + \sum_{|x'|_0 = j > i} mut(x, x') \cdot Prob(\delta > j - i) \cdot (\mathbb{E}[x'] - \mathbb{E}[x]).
 \end{aligned}$$

Let $g(k) = \min_{x \notin \mathcal{X}^*} g(x, k)$, then $g(k) > 0$. When $m > 0$, we have that $Prob(\delta \geq m)$ decreases with k by the property of Gaussian distribution, which implies that for any x , $g(x, k)$ decreases with k . Thus, $g(k)$ decreases with k , i.e., $\frac{dg(k)}{dk} \leq 0$. \square

4.3 Empirical Verification

We run (1+1)-EA on the problems to verify the theoretical results. For the (1+1)-EA, the mutation probability p is set to be $\frac{1}{n}$; for the OneMax and the Trap problems, the problem size $n = 10$ and $C = n + 1$; for the Gaussian noise, $\theta = 0$ and $\sigma = 10$. We investigate the sample size k from 1 to 100; for each k , we run the EA 1,000 times independently, where each run stops until an optimal solution is found. We use the average number of iterations and the average number of fitness evaluations as the estimation of the EFHT and the expected running time (ERT), respectively.

The results are plotted in Figures 1 and 2. For (1+1)-EA optimizing Noisy OneMax problem, Figure 1 shows that the EFHT can decrease by increasing the sample size k , however the ERT increases with k , which implies that the decrease rate of the EFHT cannot catch up with the increase rate of k . On Noisy Trap problem, we can observe from Figure 2 that both the EFHT and ERT increase with the sample size, which implies that noise is helpful and using a larger sample size reduces its positive effect. Thus, these empirical results verify our theoretical analysis.

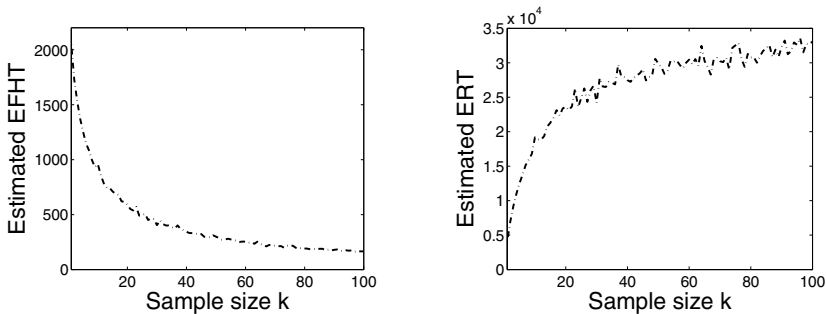


Fig. 1. Estimated EFHT and ERT for (1+1)-EA on Noisy OneMax with different sample sizes

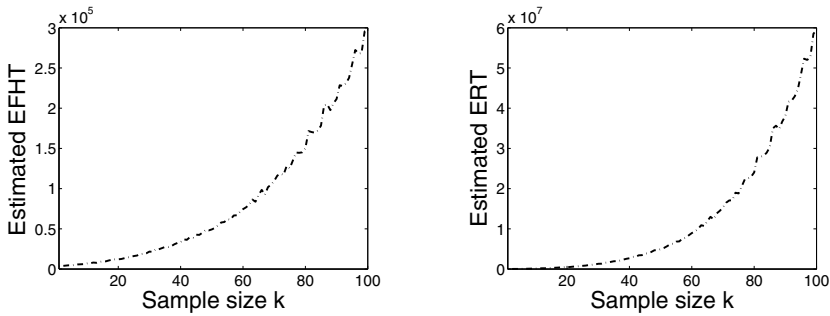


Fig. 2. Estimated EFHT and ERT for (1+1)-EA on Noisy Trap with different sample sizes

5 Conclusion

Sampling has often been employed to smooth noise in evolutionary optimization. Previous empirical studies showed conflicting results, and sampling thus has not been well understood. In this paper, we investigate its effectiveness by rigorous running time analysis. We provide a sufficient condition under which sampling is useless. Using this condition, we prove that sampling is useless for (1+1)-EA optimizing OneMax and Trap problems under additive Gaussian noise, which is also empirically verified. An intuitive interpretation of the theorems is that, noise should be removed for the OneMax problem, but the extra cost of using sampling is overwhelming; and noise should not be removed for the Trap problem, thus sampling is useless. Note that, OneMax and Trap have been recognized as the easiest and the hardest instances, respectively, in the pseudo-Boolean problem class with a unique global optimum for (1+1)-EA [21]. Thus, we conjecture that sampling might be useless for a large problem class, which will be a subject of future research. Our results on the effectiveness of sampling may guide us to design effective noise handling strategies in real optimization tasks.

References

1. Aizawa, A.N., Wah, B.W.: Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation* 2(2), 97–122 (1994)
2. Arnold, D.V.: *Noisy Optimization with Evolution Strategies*. Kluwer, Norwell (2002)
3. Auger, A., Doerr, B.: *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific, Singapore (2011)
4. Bäck, T.: *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford (1996)
5. Bartz-Beielstein, T., Blum, D., Branke, J.: Particle swarm optimization and sequential sampling in noisy environments. In: *Metaheuristics. Operations Research/Computer Science Interfaces Series*, vol. 39, pp. 261–273. Springer, Heidelberg (2007)
6. Beyer, H.G.: Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. *Computer Methods in Applied Mechanics and Engineering* 186(2), 239–267 (2000)
7. Branke, J., Schmidt, C.: Selection in the presence of noise. In: *Proceedings of the 5th ACM Conference on Genetic and Evolutionary Computation*, Chicago, IL, pp. 766–777 (2003)

8. Branke, J., Schmidt, C.: Sequential sampling in noisy environments. In: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, Birmingham, UK, pp. 202–211 (2004)
9. Branke, J., Schmidt, C., Schmeck, H.: Efficient fitness estimation in noisy environments. In: Proceedings of the 3rd ACM Conference on Genetic and Evolutionary Computation, San Francisco, CA, pp. 243–250 (2001)
10. Cantú-Paz, E.: Adaptive sampling for noisy problems. In: Proceedings of the 6th ACM Conference on Genetic and Evolutionary Computation, Seattle, WA, pp. 947–958 (2004)
11. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. *Algorithmica* 65, 224–250 (2013)
12. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. *Algorithmica* 64, 673–697 (2012)
13. Doerr, B., Hota, A., Kötzing, T.: Ants easily solve stochastic shortest path problems. In: Proceedings of the 14th ACM Conference on Genetic and Evolutionary Computation, Philadelphia, PA, pp. 17–24 (2012)
14. Droste, S.: Analysis of the (1+1) EA for a noisy OneMax. In: Proceedings of the 6th ACM Conference on Genetic and Evolutionary Computation, Seattle, WA, pp. 1088–1099 (2004)
15. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276(1-2), 51–81 (2002)
16. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence* 127(1), 57–85 (2001)
17. Iacca, G., Neri, F., Mininno, E.: Noise analysis compact differential evolution. *International Journal of Systems Science* 43(7), 1248–1267 (2012)
18. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments—a survey. *IEEE Transactions on Evolutionary Computation* 9(3), 303–317 (2005)
19. Neumann, F., Witt, C.: *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Springer, Berlin (2010)
20. Park, T., Ryu, K.R.: Accumulative sampling for noisy evolutionary multi-objective optimization. In: Proceedings of the 13th ACM Conference on Genetic and Evolutionary Computation, Dublin, Ireland, pp. 793–800 (2011)
21. Qian, C., Yu, Y., Zhou, Z.-H.: On algorithm-dependent boundary case identification for problem classes. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I. LNCS*, vol. 7491, pp. 62–71. Springer, Heidelberg (2012)
22. Sano, Y., Kita, H.: Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu, HI, pp. 360–365 (2002)
23. Siegmund, F., Ng, A.H., Deb, K.: A comparative study of dynamic resampling strategies for guided evolutionary multi-objective optimization. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, pp. 1826–1835 (2013)
24. Stagge, P.: Averaging efficiently in the presence of noise. In: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Amsterdam, The Netherlands, pp. 188–197 (1998)
25. Syberfeldt, A., Ng, A., John, R.I., Moore, P.: Evolutionary optimisation of noisy multi-objective problems using confidence-based dynamic resampling. *European Journal of Operational Research* 204(3), 533–544 (2010)
26. Yu, Y., Zhou, Z.-H.: A new approach to estimating the expected first hitting time of evolutionary algorithms. *Artificial Intelligence* 172(15), 1809–1832 (2008)
27. Zhang, Z., Xin, T.: Immune algorithm with adaptive sampling in noisy environments and its application to stochastic optimization problems. *IEEE Computational Intelligence Magazine* 2(4), 29–40 (2007)

Evolving Mixtures of n -gram Models for Sequencing and Schedule Optimization

Chung-Yao Chuang and Stephen F. Smith

The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
cychuang@cmu.edu, sfs@cs.cmu.edu

Abstract. In this paper, we describe our work on Estimation of Distribution Algorithms (EDAs) that address *sequencing problems*, i.e., the task of finding the best ordering of a set of items or an optimal schedule to perform a given set of operations. Specifically, we focus on the use of probabilistic models that are based on n -gram statistics. These models have been used extensively in modeling statistical properties of sequences. We start with an EDA that uses a bigram model, then extend this scheme to higher-order models. However, directly replacing the bigram model with a higher-order model often results in premature convergence. We give an explanation on why this is the case along with some empirical support for our intuition. Following that, we propose a technique that combines multiple models of different orders, which allows for smooth transition from lower-order models to higher-order ones. Furthermore, this technique can also be used to incorporate other heuristics and prior knowledge about the problem into the search mechanism.

1 Introduction

Estimation of Distribution Algorithms (EDAs) are a class of population-based stochastic search techniques that search the solution space by learning and sampling probabilistic models [1,2]. Using probabilistic models in the search mechanism enables EDAs to adopt techniques from machine learning and statistics to automatically discover patterns exhibited in a set of promising solutions. In past studies, EDAs have been applied to a variety of academic and real-world optimization problems [2,1], achieving competitive results in many scenarios: chemical applications [3], power systems [4], and environmental resources [5], to name a few. Most of these studies were focused on domains in which a solution can be naturally represented as a fixed-length string with no ordering dependencies. However, many interesting and important optimization problems require the determination of a best ordering of a set of items or an optimal sequence to perform a given set of operations. In this work, we are interested in solving such kind of *sequencing problems* through the paradigm of EDAs.

One classical example of this kind of problem is the Traveling Salesman Problem (TSP). The objective of the TSP is to find the shortest route for a traveling salesman who is on the mission to visit every city on a given list precisely once and then return to the initial city. This task is equivalent to finding the Hamiltonian cycle that has the smallest cost in a complete weighted graph. The TSP is

celebrated because many scientific and engineering problems can be formulated as TSPs and it has long been used to study sequencing and scheduling problems. In this paper, we will use the TSP as our model problem for evaluation purposes.

Some previous works can be found in the literature that deal with sequencing and scheduling problems by means of EDAs. However, most of these approaches are direct adaptations of EDAs designed for discrete or continuous problems that have no ordering properties. Earliest attempts [6] applied discrete EDAs as if a solution has no sequential dependencies. The obvious drawback is that the information of relative ordering among items is not explicitly considered in the constructed models. This deficiency may be the cause of low success rate in finding the global optimum as reported in [6,7,8]. Adaptation of continuous EDAs [6,9] has also been explored. Most of the research in this direction uses the random keys representation [10]. Using this representation, some of the information about the relative ordering of the items can be encoded in the probabilistic model. However, with this type of construct, an algorithm has to search for solutions in a largely redundant real-valued space. This inefficiency is reflected in their relatively inferior performance in the review by Ceberio et al. [11].

The limitation of these direct adaptation of EDAs designed for problems without ordering properties encourage the EDA community to invent other approaches that specifically target the sequencing problems. More relevant to this research is the work done by Tsutsui et al. [7,8]. They proposed an approach called Edge Histogram Based Sampling Algorithm (EHBSA), which constructs an edge histogram matrix by counting the number of occurrences that item i and item j appear consecutively in the sequences. For TSP, this is how many times the link between the i -th and j -th city is observed in promising solutions. Based on this statistics, a distribution is estimated that gives conditional probability of the next item given the previous one. This approach is equivalent to estimating a bigram model from the current population. In this research, we work on generalizing this idea to n -gram models.

Although this generalization seems straightforward, as we will show empirically, the naive approach of increasing the order of the model (e.g., using trigram instead of bigram) does not work. Instead, we developed a method that uses linear interpolation to combine multiple models of different orders, and utilize a holdout set to estimate the weight associated with each model. In this way, we can gradually shift the emphasis from a low-order model to higher order ones as longer patterns emerge in the population. Furthermore, as we will show in Sect. 5, this technique can also be used to incorporate other heuristics and prior knowledge about the problem into the search mechanism.

In the next section, we will describe the formulation of the n -gram models. After that, Sect. 3 introduces our approach of using n -gram models for guiding the search process. It also discusses the difficulty encountered when moving from bigram model to higher-order models. In Sect. 4, we present a method that is able to combine multiple models of different orders, and thus provides a smooth transition from lower-order model to higher-order ones. Sect. 5 further describes how we can use this same method to incorporate other heuristics and prior knowledge about the problem into the search mechanism. We briefly discuss some characteristics of our proposal in Sect. 6. Finally, Sect. 7 summarizes this paper and points out the future direction of our work.

2 Modeling Sequence Properties with n -gram Statistics

An n -gram is a pattern of n consecutive items, which is usually a segment from a longer sequence. Such a construct is often used in the tasks of modeling statistical properties of sequences, especially in the field of natural language processing (NLP). For example, a classic task in NLP is to predict the next word given the previous words. Such task can be stated as attempting to estimate the conditional probability of observing some item w_i as the next item given the history of items seen so far. The n -gram approach to this estimate is to make a Markov assumption that only prior local context—the last few items—affects the next item. More formally, we are interested in estimating

$$P(W_i = w_i | W_{i-n+1} = w_{i-n+1}, \dots, W_{i-1} = w_{i-1})$$

where the sequence w_1, w_2, \dots is some instantiation of a sequence of random variables W_1, W_2, \dots . In the following, we will use $P(w_i | w_{i-n+1} \dots w_{i-1})$ as a shorthand for this probability function.

The obvious first answer to the above formulation is to suggest using a *maximum likelihood estimate* (MLE):

$$P_{\text{MLE}}(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_{i-1} w_i)}{\sum_{v \in \mathbf{V}} C(w_{i-n+1} \dots w_{i-1} v)}$$

where $C(w_{i-n+1} \dots w_{i-1} w_i)$ is the frequency of a certain n -gram in the training samples, and \mathbf{V} is the set of possible items. However, a drawback is that MLE assigns a zero probability to unseen events, which effectively zeros out the probability of sequences with component n -grams that just happened not appearing in the training samples. For our scenario, this creates a risk of arbitrarily discarding some portion of the unexplored search space. Thus, we need a more suitable estimator that takes previously unseen patterns into consideration.

A simple solution to this problem is to smooth the distribution with some *pseudocount* κ :

$$P_{\kappa}(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{C(w_{i-n+1} \dots w_{i-1} w_i) + \kappa}{\sum_{v \in \mathbf{V}} (C(w_{i-n+1} \dots w_{i-1} v) + \kappa)}$$

where κ is usually set to a value smaller than 1. In this work, we use this simple method to allocate probability mass for unobserved events, though more sophisticated estimators are possible for this task.

3 An EDA Framework with n -gram Models

This section describes the basic approach that uses n -gram models in the EDA framework for sequencing and scheduling problems. To briefly recap the operations performed by an EDA: At each iteration, we start with a set of promising solutions, then the algorithm constructs a probabilistic model based on the statistics gathered from those solutions. Once a model is learned, a number of new solutions will be generated by sampling the model to replace solutions in the current population according to some replacement strategy.

Table 1. Observations on solving **gr48**

Method	Success Rate	# of Evaluations	
		mean	std
2G	30/30	277024	34535.4
3G	9/30	224640	118490.2
2G $\xrightarrow{800 \text{ iter.}}$ 3G	30/30	240032	20753.3

In this work, instead of generating an entire solution anew, we first take an existing solution from the current population and randomly extract a subsequence from that solution. This segment will then be taken as the first part of the new solution and serve as the “history” on which the further sampling is based. This kind of *partial sampling* technique has been used by previous researchers such as Chuang and Chen [12] and Tsutsui et al. [8], achieving better usage of diversity and resulting in significant improvement in performance. For our purpose, this has an additional benefit of providing a convenient basis to initialize the sampling from the n -gram models.

Once we have the first part of the new solution, we will further generate the rest of the solution by repeatedly sampling the n -gram model, with previous $n - 1$ items as the history. In order to have a valid solution, the set of possible items \mathbf{V} may be varied as the sampling goes on. For example, when dealing with the TSPs, the set of possible next cities \mathbf{V} has to be altered to exclude cities that have already been included in the constructed partial solution.

To summarize the overall flow of the algorithm: At each iteration, we slide a window of size n through each solution in the current population to obtain the frequency counts of n -gram patterns. This statistics is then used for estimating an n -gram model in the form of a conditional distribution. To generate a new solution, we use partial sampling on an existing solution in the current population. Each solution in the current population is visited once for such sampling. Following each partial sampling, a replacement competition is hold between the new solution and the solution from which that new solution’s starting segment was extracted. Note that we use replacement as the sole means for selecting promising solutions, i.e., better solutions are preserved under the replacement process. This is similar to the evolution strategies [13], in which every solution in the current population is seen as a potentially good solution because they have survived previous replacement competitions.

As a first step, we examine the performance of using a bigram model

$$P_{2G}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) + \kappa}{\sum_{v \in \mathbf{V}} (C(w_{i-1}v) + \kappa)}$$

for solving a 48-city TSP instance, **gr48**, taken from TSPLIB¹ [14]. Let ℓ denote the problem size. In this experiment, the population size N is set to 5ℓ , the pseudocount is set to $\kappa = 0.01$, and the termination criterion is when either the optimal tour is found or when the algorithm reaches 50ℓ iterations. We ran the algorithm 30 times to observe the average performance. The result of the experiment is presented in Table 1. It shows the success rate in finding the optimal tour and the average number of objective function evaluations used by

¹ <http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/>

the algorithm among the successful runs. It can be seen that the bigram approach gives a pretty decent performance. It shows a high success rate in finding the global optimum, and a reasonable usage of function evaluations.

A tempting thought to proceed is to increase the order of the model. For example, instead of using bigram, we could use a trigram model

$$P_{3G}(w_i|w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i) + \kappa}{\sum_{v \in \mathbf{V}} (C(w_{i-2}w_{i-1}v) + \kappa)}$$

for learning the patterns. However, as shown in Table 1, this results in a significant drop in success rate. Our explanation is that at early stage of a run, there are not so many long patterns that are of good quality. If we attempt to use a higher-order model to learn longer patterns when there are none, we will end up encoding mediocre patterns into the model. Thus, a better way to proceed may be to use a low-order model like bigram model at the beginning of a run and switch to higher-order ones after longer patterns have emerged in the population.

To provide some empirical support for this conjecture, we modified the process to begin with a bigram model, and then switch to the trigram after 800 iterations. As shown in the third row of Table 1, the success rate returns to the same level as using the bigram model and it shows some improvement on function evaluations over the bigram approach. It seems that we can use this technique to gradually move to higher-order models. However, choosing an appropriate schedule to make such switches is a nontrivial task. To address this issue and avoid having to choose a fixed switching point to elevate to a higher-order model, we propose an approach that estimates multiple n -gram models of different orders and combines those models into one composite model. The method automatically calibrates the degree of emphasis placed on each n -gram model. The detail of our formulation is presented in the next section.

4 Combining Multiple Models with Linear Interpolation

Specifically, we formulate the synthesis of multiple models as a linear combination of distributions

$$P(w_i|\mathbf{h}_i) = \sum_j \lambda_j P_j(w_i|\mathbf{h}_i) \quad (1)$$

where \mathbf{h}_i represents the history of items we have seen so far, j is the index to a particular model, and λ_j is the weight associated with the j -th model such that $\lambda_j > 0$ and $\sum_j \lambda_j = 1$. A combination of bigram and trigram model will be

$$P_{2G+3G}(w_i|\mathbf{h}_i) = \lambda_{2G} P_{2G}(w_i|w_{i-1}) + \lambda_{3G} P_{3G}(w_i|w_{i-2}w_{i-1})$$

Assuming that we want to combine K models together, for this formulation, we have to determine K weights, $\lambda_1, \lambda_2, \dots, \lambda_K$, associated with those models. To do this, we reserve a portion of the population for the task of estimating appropriate values for those λ_j 's. Suppose that there are M items in such a holdout set for which we can give conditional probabilities. For each model P_j , we create a probability stream $\mathbf{p}_j = (p_{j1}, p_{j2}, \dots, p_{jM})$ where p_{ji} is the probability of item w_i predicted by the model P_j , i.e., $p_{ji} = P_j(w_i|\mathbf{h}_i)$. These K probability

Algorithm 1. Estimating the Weights λ_j 's

Input: A set of probability streams $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_K\}$,
such that each $\mathbf{p}_j = (p_{j1}, p_{j2}, \dots, p_{jM})$ is of length M .

Initialize $\mathbf{\Lambda}^{(0)} = \{\lambda_1^{(0)}, \lambda_2^{(0)}, \dots, \lambda_K^{(0)}\}$ s.t. each $\lambda_j^{(0)} > 0$ and $\sum_{j=1}^K \lambda_j^{(0)} = 1$
repeat from $t = 0$

$$\text{For } j = 1 \dots K, \text{ update } \lambda_j \text{ using } \lambda_j^{(t+1)} = \frac{1}{M} \sum_{i=1}^M \frac{\lambda_j^{(t)} p_{ji}}{\sum_{k=1}^K \lambda_k^{(t)} p_{ki}}$$

$t = t + 1$

until the difference between $\mathbf{\Lambda}^{(t)}$ and $\mathbf{\Lambda}^{(t-1)}$ is small

return $\mathbf{\Lambda}^{(t)}$

streams (each of length M) are then used as the input to Algorithm 1 [15]. The resulting λ_j 's optimize the average likelihood with respect to this holdout set².

To illustrate the search behavior, Fig. 1 shows the variation of weights in a typical run that uses a combination of the bigram and trigram models. The weight for the bigram model starts out with a high value and gradually decreases. On the other hand, the weight of the trigram model will begin to dominate in later part of the search, meaning that we do more and more sampling with the trigram model. Based on this self-adaptive behavior, which adjusts the weights automatically, we call our proposal the ‘‘evolving mixture.’’

To evaluate our proposal, we performed a set of experiments on ten TSP instances from TSPLIB. The parameters to the algorithms are listed in Table 2. As before we ran each algorithm 30 times to give the average performance. The outcomes are presented in Table 3. In each table, we listed the success rate of each algorithm and its average usage of function evaluations among the successful runs. Note that the trailing number in each instance’s name represents the number of cities in that instance.

The result of using evolving mixture to combine bigram and trigram is listed in the third row of each table. Comparing to the trigram approach (second row of the table), it gives a significantly better success rate, which is at the same level of the bigram approach. On the other hand, when compared with bigram approach, the evolving mixture approach uses less function evaluations on average.

5 Incorporating Other Heuristics

In its formulation of Eq. (1), we did not put a restriction on the type of the models that can be included. It does not have to be an n -gram model for the

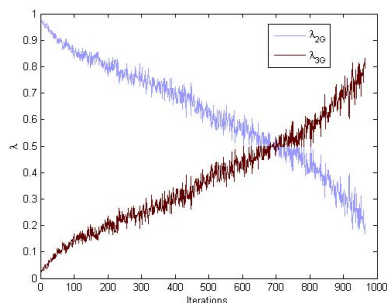


Fig. 1. Typical variation of weights associated with P_{2G} and P_{3G} . This illustration is from a run on **gr48** with $N = 450$ and 30% of the population as holdout set.

² For simplicity, we use $\max_j |\lambda_j^{(t)} - \lambda_j^{(t-1)}| < 0.001$ as condition to terminate Algo. 1.

Table 2. Parameter Settings

(a) For problem size $\ell < 70$		(b) For problem size $\ell \geq 70$	
Parameters	Value	Parameters	Value
population size	$N = 5\ell$	population size	$N = 5\ell$
pseudocount	$\kappa = 0.01$	pseudocount	$\kappa = 0.001$
size of holdout set	$R = \lfloor \frac{N}{10} \rfloor$	size of holdout set	$R = \lfloor \frac{N}{10} \rfloor$
max iterations	50ℓ	max iterations	80ℓ

evolving mixture to work. The only constraint is that the model takes the form of a (conditional) probability distribution. We can even push the envelop to include something that just *looks like* a probability distribution.

For example, if we want to incorporate a distance-based heuristic for TSP into the search mechanism, we could do so by crafting an “artificial distribution”

$$P_{\text{DH}}(w_i|w_{i-1}) = \frac{d(w_{i-1}, w_i)^{-10}}{\sum_{v \in \mathbf{V}} d(w_{i-1}, v)^{-10}}$$

where $d(u, v)$ is the distance between city u and city v . In short, this formula assigns a larger probability mass to a city that has shorter link to the last city in the partial tour constructed so far.

To see the effect of incorporating such a heuristic, we performed experiments on the same set of TSP instances as previous section. The results are presented in the fourth rows of Table 3. It can be observed that the performance improves significantly. Comparing to using only n -gram models, it uses far less function evaluations, especially for larger instances, while retaining a high success rate. For completeness, we also include the results of using solely the distance-based heuristic for updating the population.

6 Discussion

The previous section showcased how we can use evolving mixture to incorporate other heuristics and prior knowledge about the problem into the search mechanism. The results also provide some support for our intuition that different methods or heuristics may be more suitable than others at different stages of the optimization process. For example, Fig. 2 shows the shifts of weights among three distributions in a run for solving `st70`. It illustrates how the evolving mixture adjusts the emphasis on different models and heuristics at different stages of the process. This adjustment is dynamically determined based on the promising solutions in the holdout set. We believe that this dynamic behavior may lead to a more efficient search strategy for finding the global optima.

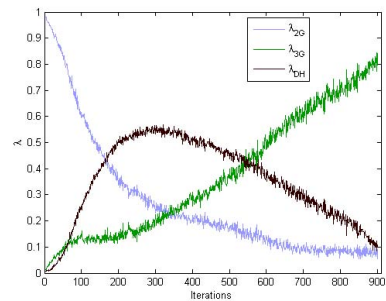


Fig. 2. Typical variation of weights associated with P_{2G} , P_{3G} and P_{DH} . This illustration is from a run on `st70` with $N = 500$ and 30% of the population as holdout set.

Table 3. Experiment Results

(a) ulysses16				(b) gr24			
Method	Success Rate	# of Evaluations		Method	Success Rate	# of Evaluations	
		mean	std			mean	std
2G	30/30	5461.3	902.4	2G	30/30	18436.0	2356.7
3G	23/30	7207.0	10552.4	3G	25/30	16574.4	11117.6
2G+3G	30/30	5040.0	1250.3	2G+3G	30/30	16492.0	2325.1
2G+3G+DH	30/30	4597.3	973.0	2G+3G+DH	30/30	11432.0	1717.7
DH	30/30	17725.3	12070.3	DH	30/30	13340.0	6265.2

(c) bay29				(d) att48			
Method	Success Rate	# of Evaluations		Method	Success Rate	# of Evaluations	
		mean	std			mean	std
2G	30/30	36332.0	3914.3	2G	27/30	298053.3	54469.6
3G	22/30	40336.4	43365.6	3G	10/30	196176.0	127370.7
2G+3G	30/30	31218.5	3409.1	2G+3G	26/30	215021.5	18175.5
2G+3G+DH	29/30	22785.0	3251.0	2G+3G+DH	30/30	111544.0	10272.6
DH	19/30	107063.4	44051.6	DH	28/30	265637.1	90849.8

(e) gr48				(f) ei151			
Method	Success Rate	# of Evaluations		Method	Success Rate	# of Evaluations	
		mean	std			mean	std
2G	30/30	277024.0	34535.4	2G	19/30	493572.6	73781.0
3G	9/30	224640.0	118490.2	3G	2/30	199665.0	39308.1
2G+3G	30/30	230048.0	23985.8	2G+3G	21/30	391182.1	90013.1
2G+3G+DH	30/30	154984.0	20243.5	2G+3G+DH	29/30	217031.4	52819.1
DH	0/30	N/A	N/A	DH	8/30	448513.1	157609.4

(g) berlin52				(h) st70			
Method	Success Rate	# of Evaluations		Method	Success Rate	# of Evaluations	
		mean	std			mean	std
2G	29/30	264276.6	74339.8	2G	30/30	1426938.3	201534.3
3G	9/30	320348.9	237082.3	3G	12/30	554983.3	69040.0
2G+3G	29/30	217171.7	26798.3	2G+3G	29/30	930444.8	76827.8
2G+3G+DH	30/30	113906.0	9854.6	2G+3G+DH	30/30	298841.7	16886.3
DH	30/30	180882.0	77478.7	DH	0/30	N/A	N/A

(i) kroA100				(j) lin105			
Method	Success Rate	# of Evaluations		Method	Success Rate	# of Evaluations	
		mean	std			mean	std
2G	30/30	3484900.0	208750.7	2G	30/30	3772142.5	230364.5
3G	7/30	2068357.1	233884.2	3G	3/30	1984325.0	154667.8
2G+3G	30/30	2773483.3	107248.1	2G+3G	28/30	2844131.3	119353.2
2G+3G+DH	30/30	650783.3	46476.1	2G+3G+DH	30/30	572197.5	19629.8
DH	0/30	N/A	N/A	DH	0/30	N/A	N/A

Table 4. Performance Comparison

(a) Solving gr48					(b) Solving pr76				
Method	N	Success Rate	# of Evaluations		Method	N	Success Rate	# of Evaluations	
			mean	std				mean	std
OX	120	0/10	N/A	N/A	OX	120	0/10	N/A	N/A
OX	960	2/10	287852	6706	OX	960	0/10	N/A	N/A
eER	120	0/10	N/A	N/A	eER	120	0/10	N/A	N/A
eER	960	5/10	166286	4932	eER	960	3/10	394887	22321
PMX	120	0/10	N/A	N/A	PMX	120	0/10	N/A	N/A
PMX	960	0/10	N/A	N/A	PMX	960	0/10	N/A	N/A
EHBSA-WT	120	10/10	144032	29115	EHBSA-WT	120	9/10	457147	65821
2G+3G	120	10/10	131724	16748	2G+3G	120	10/10	405660	54893
2G+3G+DH	120	10/10	83508	17105	2G+3G+DH	120	10/10	195960	28123

Readers who are familiar with the Ant Colony Optimization (ACO) [16] might relate ACO to our approach in the aspect that they both use some distance-based heuristics to provide partial guidance for the search process. However, the crucial difference between the two is that ACO holds the heuristic term constant throughout a run with a user-specified parameter. On the other hand, our proposal dynamically adjusts the weights associated with the incorporated models and heuristics as the search proceeds. As mentioned previously, we think this dynamic adjustment may be more efficient than the static combination. Furthermore, it also simplifies some of the manual tuning associated with incorporating multiple models and heuristics.

7 Summary and Future Works

In this work, we have experimented with a set of EDAs that use probabilistic models estimated from n -gram statistics. To provide a smooth transition from lower-order model to higher-order ones, we proposed using a method that combines multiple models in the form of a linear combination. The weights associated with those models are estimated automatically from a reserved portion of the population. An additional advantage of this approach is that it also provides a convenient way to incorporate other heuristics and prior knowledge about the problem into the search mechanism.

As future work, we would like to compare our proposal to other approaches that also deal with sequencing problems. To provide some initial comparison, we adopt some experiment results from Tsutsui et al.’s work [8] which lists the performance of their method, EHBSA-WT³, along with three other classical EA approaches, OX [17], eER [18] and PMX [19], on two TSP instances taken from TSPLIB. Note that EHBSA-WT gave the best empirical performance on TSPs in the review by Ceberio et al. [11]. To compare our proposals with those approaches, we adopted their settings for running our algorithms. Table 4 shows the results of our methods along with the data taken from [8]. Note that in these experiments, we followed their setting which only performs ten runs per algorithm. For statistical significance, this might not be enough. So, the comparison should be taken merely as suggestive. However, the margin between our proposal and other methods seems to be quite prominent. Thus, we believe our proposal offers a promising direction for further investigation and development.

³ A variant of EHBSA. More specifically, we adopt the results of configuration EHBSA-WT2, which is more similar to our proposal in the way of doing partial sampling.

References

1. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers (2001)
2. Pelikan, M., Sastry, K., Cantú-Paz, E.: Scalable optimization via probabilistic modeling: From algorithms to applications. Springer (2006)
3. Mendiburu, A., Miguel-Alonso, J., Lozano, J.A., Ostra, M., Ubide, C.: Parallel EDAs to create multivariate calibration models for quantitative chemical applications. *Journal of Parallel and Distributed Computing* 66(8), 1002–1013 (2006)
4. Chen, C.H., Chen, Y.P.: Real-coded ECGA for economic dispatch. In: Proceedings of the 9th GECCO, pp. 1920–1927. ACM (2007)
5. Ducheyne, E.I., De Baets, B., De Wulf, R.: Probabilistic models for linkage learning in forest management. In: Knowledge Incorporation in Evolutionary Computation, pp. 177–194. Springer (2005)
6. Robles, V., de Miguel, P., Larrañaga, P.: Solving the traveling salesman problem with EDAs. In: Estimation of Distribution Algorithms, pp. 211–229. Springer (2002)
7. Tsutsui, S.: Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram. 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. 224–760. Springer, Heidelberg (2002)
8. Tsutsui, S., Pelikan, M., Goldberg, D.E.: Using edge histogram models to solve permutation problems with probabilistic model-building genetic algorithms. Technical Report 2003022 (2003)
9. Lozano, J.A., Mendiburu, A.: Solving job scheduling with estimation of distribution algorithms. In: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, pp. 231–242. Kluwer Academic Publishers (2002)
10. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. *INFORMS Journal on Computing* 6(2), 154–160 (1994)
11. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence* 1(1), 103–117 (2012)
12. Chuang, C.Y., Chen, Y.P.: On the effectiveness of distributions estimated by probabilistic model building. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 391–398. ACM (2008)
13. Beyer, H.G., Schwefel, H.P.: Evolution strategies—a comprehensive introduction. *Natural Computing* 1(1), 3–52 (2002)
14. Reinelt, G.: TSPLIB—a traveling salesman problem library. *ORSA Journal on Computing* 3(4), 376–384 (1991)
15. Jelinek, F., Mercer, R.L.: Interpolated estimation of markov source parameters from sparse data. In: Proceedings of the Workshop on Pattern Recognition in Practice (1980)
16. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1), 53–66 (1997)
17. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the traveling salesman problem. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application, pp. 224–230. Erlbaum Associates Inc., Hillsdale (1987)
18. Starkweather, T., McDaniel, S., Whitley, D., Mathias, K., Whitley, C.: A comparison of genetic sequencing operators. In: Proceedings of the Fourth International Conference on Genetic Algorithms (1991)
19. Goldberg, D.E., Lingle, R.: Alleles, loci and the traveling salesman problem. In: Proceedings of the First International Conference on Genetic Algorithms and Their Applications, pp. 154–159 (1985)

A Study on Multimemetic Estimation of Distribution Algorithms

Rafael Nogueras and Carlos Cotta

Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga,
ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain
ccottap@lcc.uma.es

Abstract. Multimemetic algorithms (MMAs) are memetic algorithms in which memes (interpreted as non-genetic expressions of problem-solving strategies) are explicitly represented and evolved alongside genotypes. This process is commonly approached using the standard genetic procedures of recombination and mutation to manipulate directly information at the memetic level. We consider an alternative approach based on the use of estimation of distribution algorithms to carry on this self-adaptive memetic optimization process. We study the application of different EDAs to this end, and provide an extensive experimental evaluation. It is shown that elitism is essential to achieve top performance, and that elitist versions of multimemetic EDAs using bivariate probabilistic models are capable of outperforming genetic MMAs.

1 Introduction

Memetic algorithms [8, 14] can be regarded as a pragmatic integration of ideas from population-based global search techniques and trajectory-based local search techniques [12]. One of the central tenets in the paradigm is the notion of *meme*, famously defined by Richard Dawkins as units of imitation [5]. Within this optimization context, memes translate to computational problem-solving procedures. While this definition is broad enough to encompass a wide variety of techniques, it is typically the case that memes are assimilated to local-search procedures. Even more so, these procedures are often fixed or pre-defined and therefore the MA can be regarded as operating with static implicit memes. This said, the explicit management of memes is a topic that has been around for some time now, and can be found in, e.g., multimemetic algorithms (MMAs) [9], in which each solution carries memes indicating how self-improvement is going to be conducted. Such memes are subject to evolution using the standard genetic procedures of recombination and mutation, thus conforming a self-adaptive search approach.

In this work we are going to consider the use of estimation of distribution algorithms (EDAs) [10, 11, 17] as the underlying search engine for multimemetic optimization. While the use of local search procedures in combination with EDAs is a widely-known approach to inject problem-dependent knowledge and improve the efficiency of the optimization process –see, e.g., [16, 18, 19, 22]– the use of EDAs for self-adaptive memetic optimization has been less explored. The contribution

of this work is taking some steps in this direction, studying different approaches for the application of EDAs to multimemetic optimization, and providing an extensive empirical evaluation of their performance.

2 Multimemetic EDAs

As mentioned above, the core idea of MMAs is the explicit treatment of memes within the evolutionary process. Hence, we shall firstly describe the representation of memes, before getting into the deployment of EDAs in this context.

2.1 Meme Representation and Application

Memes, conceived as non-genetic expressions of problem-solving strategies, can be represented in many ways depending on the level of abstraction and problem dependance considered. In this work we follow some ideas posed by Smith [20] in the context of pseudoboolean function optimization. Therein, memes are expressed as pattern-based rewriting rules [*condition*→*action*] as follows: let $[C \rightarrow A]$ be a meme, where $C, A \in \Sigma^r$ with $\Sigma = \{0, 1, \#\}$ and $r \in \mathbb{N}$ being some constant. In this ternary alphabet ‘#’ represents a wildcard symbol; let $g_1 \cdots g_n$ be a genotype; a meme $[c_1 \cdots c_r \rightarrow a_1 \cdots a_r]$ could be applied on any substring of the genotype into which the condition fits, i.e., for which $g_i \cdots g_{i+r-1} = c_1 \cdots c_r$ (for this purpose, wildcard symbols in the condition are assumed to match any symbol in the genotype). If the meme were to be applied on position i , its action would be to implant the action $a_1 \cdots a_r$ in that part of the genotype, i.e., letting $g_i \cdots g_{i+r-1} \leftarrow a_1 \cdots a_r$ (in this case, wildcard symbols in the action are taken as don’t-change symbols, that is, keeping unchanged the corresponding symbol in the genotype – we depart here from [20] in which wildcards in the action represented the binary complement of the original gene). The genotype is scanned in a randomized order to check for potential application sites of the meme so as to avoid positional bias. If a match is found the meme is applied and the resulting neighboring genotype is evaluated. A parameter w determining the maximal number of meme applications per individual is used to keep the total cost of the process under control. The best neighbor generated throughout the process is kept if it is better than the current genotype.

2.2 EDA Approaches

The underlying idea in the MMA model considered is to have genetic and memetic information linked within a single individual, i.e., each individual carries a genotype and a meme. Once an individual is generated, its genotype is evaluated and the meme is subsequently applied in order to improve it. Whereas these individuals are generated by means of evolutionary operators –such as recombination and mutation– in standard genetic MMAs, multimemetic EDAs approach this by probabilistic sampling of a certain distribution which is evolved during the run. Let us focus on how this is done.

EDAs try to learn the joint probability distribution $p(\mathbf{x})$ representing the most promising individuals at each generation. Such generations comprise a cycle of (1) sampling $p(\mathbf{x})$ to obtain a population pop , selecting the most promising individuals pop' from pop , and updating $p(\mathbf{x})$ using pop' . Different EDAs can be considered depending on the way they model the joint probability distribution $p(\mathbf{x})$. In this work, we have considered the following ones:

- *Univariate models*: variables are assumed to be independent and hence the joint probability distribution $p(\mathbf{x})$ is factorized as

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i).$$

The simplest such EDA is UMDA [13], in which $p(x_i)$ is estimated as

$$p(x_i = v) = \frac{1}{k} \sum_{j=1}^k \delta(pop'_{ji}, v),$$

where $k = |pop'|$, pop'_{ji} is the value of the i -th variable of the j -th individual in pop' , and $\delta(\cdot, \cdot)$ is Kronecker delta ($\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise). A generalization of UMDA is PBIL [2], an algorithm in which the probabilistic model is updated using a linear combination of its current value and the new value learnt from the sample, i.e.,

$$p'(x_i = v) = (1 - \eta)p(x_i = v) + \eta \frac{1}{k} \sum_{j=1}^k \delta(pop'_{ji}, v)$$

for some learning rate parameter η ($0 < \eta \leq 1$). Note that PBIL reduces to UMDA for $\eta = 1$.

- *Bivariate models*: these models can capture low order dependencies by assuming relations between pairs of variables. More precisely, in the models considered here $p(\mathbf{x})$ is factorized as

$$p(\mathbf{x}) = p(x_{i_1}) \prod_{j=2}^n p(x_{i_j} | x_{i_{a(j)}}),$$

where $i_1 \cdots i_n$ is a permutation of the indices $1 \cdots n$, and $a(j) < j$ is the permutation index of the variable which x_{i_j} depends on. The particular EDAs considered within this class are based on MIMIC [3] and COMIT [1]. In the first case, we assume $a(j) = j - 1$ (i.e., each variable depends on the previous one in the permutation) and the permutation is built by picking i_1 as the variable with the lowest entropy $H(X_k)$ in the selected sample pop' , and then picking i_j ($j > 1$) as the variable (among those not yet selected) that minimizes the conditional entropy $H(X_k | X_{i_{j-1}})$. Along this line, we build a COMIT-based approach by not restricting $a(j) = j - 1$, thus picking

i_j as the variable that minimizes $H(X_k|X_{i_s}, s < j)$. Thus, while in the first case we have a linear dependence structure, in this second case we have a tree dependence structure. Note finally that in these bivariate multimemetic EDAs we compute separate models for both genotypes and memes.

In all cases, the probability estimation for model updating includes Laplace correction [4] in order to prevent premature convergence, always allowing a non-zero rate of exploration. Furthermore, we also consider for each of the EDAs presented an elitist counterpart¹, in which the new population is created by truncation selection from the union of the selected sample in the previous step and the sample extracted from the current model. As shown by [7], Laplace-corrected elitist EDAs can converge to a population containing the global optimum.

3 Experimental Analysis

In order to analyze the performance of the multimemetic EDAs described in previous section we have considered a collection of pseudoboolean optimization problems. These are described in Sect. 3.1; subsequently we shall analyze the results in Sect. 3.2.

3.1 Benchmark and Settings

The test suite comprises four different problems defined on binary strings, namely Deb's trap function [6], Watson et al.'s hierarchically consistent test problems (HIFF and HXOR) [21] and Boolean satisfiability. These are described below.

Deb's 4-bit fully deceptive function (TRAP henceforth) is defined as

$$f_{trap}(b_1 \cdots b_4) = \begin{cases} 0.6 - 0.2 \cdot u(b_1 \cdots b_4) & \text{if } u(b_1 \cdots b_4) < 4 \\ 1 & \text{if } u(b_1 \cdots b_4) = 4 \end{cases} \quad (1)$$

where $u(s_1 \cdots s_i) = \sum_j s_j$ is the unitation (number of 1s in a binary string) function. This function is used as the basic block to build a higher-order problem by concatenating k such blocks, and defining the fitness of a $4k$ -bit string as the sum of the fitness contribution of each block. In our experiments we have considered $k = 32$ subproblems (i.e., 128-bit strings, $opt = 32$).

As to the hierarchically consistent test problems, they are recursive epistatic functions defined for binary strings of 2^k bits by means of two auxiliary functions $f : \{0, 1, \times\} \rightarrow \{0, 1\}$ (used to score the contribution of building blocks), and $t : \{0, 1, \times\} \rightarrow \{0, 1, \bullet\}$ (used to capture the interaction of building blocks), where ' \bullet ' is used as a *null* value. In the case of the Hierarchical if-and-only-if (HIFF) function f and t are defined as:

¹ Note that the original definition of COMIT was already intrinsically elitist. Here we consider both an elitist and a non-elitist version of this approach.

$$f(a, b) = \begin{cases} 1 & a = b \neq \bullet \\ 0 & \text{otherwise} \end{cases} \quad t(a, b) = \begin{cases} a & a = b \\ \bullet & \text{otherwise} \end{cases}$$

We use these two functions as follows:

$$\text{HIFF}_k(b_1 \cdots b_n) = \sum_{i=1}^{n/2} f(b_{2i-1}, b_{2i}) + 2 \cdot \text{HIFF}_{k-1}(b'_1, \dots, b'_{n/2}) \quad (2)$$

where $b'_i = t(b_{2i-1}, b_{2i})$ and $\text{HIFF}_0(\cdot) = 1$. The Hierarchical XOR (HXOR) works similarly but changing f so as to provide a fitness contribution of 1 when $a = 1$ and $b = 0$ or vice versa, and having in that case $t(a, b) = a$ (and $t(a, b) = \times$ otherwise). We have considered $k = 7$ (i.e., 128-bit strings, $\text{opt} = 576$)

Finally, the Boolean satisfiability problem is a classical NP-complete problem which amounts to finding a truth assignment to n variables such that a certain Boolean formula Φ is satisfied. We consider this formula is expressed in conjunctive normal form with $n = 128$ variables and $k = 3$ variables per clause; this problem is known to have an easy-hard-easy phase transition when varying the ratio m/n . The difficulty peak is located around $m = 4.3n$. We use a problem generator approach in this case, generating a different satisfiable instance with this critical clauses/variable ratio ($\text{opt} = m = 550$) in each run of the MMA.

We consider multimemetic EDAs as described in Sect. 2, with a population size of $\mu = 128$ individuals. Selection is done by truncation, keeping the best 50% individuals to update the probabilistic model, and a learning rate $\eta = 0.1$ is used in PBIL. The memes are expressed as rules of length $r = 3$ and we consider $w = 1$ (one rewriting is done and kept if the solution is improved). In all cases the cost of applying a meme is accounted as a fractional evaluation (i.e., as the fraction of the fitness function that needs being recomputed due to genotypic changes) and added to the total number of evaluations. A run is terminated upon reaching 50,000 evaluations, and 20 runs are performed for each problem and algorithm. To gauge the results, we also include in the experimentation an equivalent evolutionary MMA –termed sMMA henceforth– using the same population size ($\mu = 128$) and a generational reproductive plan with binary tournament for parent selection, one-point crossover ($p_X = 1.0$), bit-flip mutation ($p_M = 1/\ell$, where $\ell = 128$ is the number of bits), local-search (conducted using the meme linked to the individual) and replacement of the worst parent (an inherently elitist strategy, following the model presented in [15] – previous experiments with a non-elitist sMMA yielded globally inferior results, and so did a MA using a fixed bit-flip meme). In this sMMA, offspring inherit the meme of the best parent, which is then subject to mutation with probability p_M .

3.2 Experimental Results

Full numerical results are provided in Table 1. Firstly, notice that elitist versions of multimemetic EDAs (denoted by a subscript e) perform in general much better than their non-elitist counterparts. Furthermore, while the latter are in most cases inferior to the sMMA, elitist multimemetic EDAs provide top performance

Table 1. Results (20 runs) of the different MMAs on TRAP, HIFF, HXOR and SAT. The median (\tilde{x}), mean (\bar{x}) and standard error of the mean ($\sigma_{\bar{x}}$) are shown.

	TRAP		HIFF		HXOR		SAT	
	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$
sMMA	31.4	30.0 \pm 0.5	408.0	427.6 \pm 13.9	360.0	360.2 \pm 4.4	547.0	546.6 \pm 0.4
UMDA	20.0	20.5 \pm 0.2	363.0	385.9 \pm 19.5	320.5	324.6 \pm 5.2	548.0	548.0 \pm 0.3
UMDA _e	23.9	25.2 \pm 0.8	576.0	517.5 \pm 16.9	348.0	341.6 \pm 6.9	548.0	548.3 \pm 0.2
PBIL	19.2	19.2 \pm 0.0	276.5	275.1 \pm 3.4	270.5	271.0 \pm 2.4	543.0	543.8 \pm 0.5
PBIL _e	24.6	25.7 \pm 0.7	441.5	441.7 \pm 12.1	259.0	258.6 \pm 2.2	548.5	548.3 \pm 0.3
MIMIC	20.3	20.2 \pm 0.1	328.5	330.3 \pm 5.0	310.0	312.9 \pm 2.5	546.0	545.7 \pm 0.4
MIMIC _e	31.6	31.3 \pm 0.2	472.0	493.6 \pm 16.3	393.5	397.6 \pm 4.7	548.0	548.2 \pm 0.2
COMIT	21.0	21.0 \pm 0.2	337.5	342.9 \pm 5.9	330.0	328.1 \pm 3.7	548.0	548.0 \pm 0.2
COMIT _e	32.0	32.0 \pm 0.0	424.0	443.8 \pm 12.2	408.0	419.4 \pm 7.8	548.0	548.0 \pm 0.3

Table 2. Statistical comparison among the different multimemetic EDAs using Wilcoxon ranksum ($\alpha = 0.05$). For each problem/EDA three symbols are provided, respectively indicating how the algorithm compares with its (non-)elitist counterpart, with sMMA, and with the algorithm with the highest median for the corresponding problem (which is marked with a star \star in this third position). A white/black circle (\circ/\bullet) indicates the algorithm labeled in the column has a worse/better median with statistical significance. A ‘=’ sign indicates no statistically-significant difference.

	UMDA	UMDA _e	PBIL	PBIL _e	MIMIC	MIMIC _e	COMIT	COMIT _e
TRAP	ooo	●oo	ooo	●oo	ooo	●=o	ooo	●●★
HIFF	ooo	●●★	ooo	●=o	ooo	●●=	ooo	●=o
HXOR	ooo	●oo	●oo	ooo	ooo	●●o	ooo	●●★
SAT	=●=	=●=	ooo	●●★	o=o	●●=	=●=	=●=

in all problems, and are also globally superior to non-memetic EDAs (results not shown). This is further investigated in Table 2. As it can be seen, the superiority of elitist algorithms over non-elitist ones is statistically significant in all cases, except in PBIL for HXOR and UMDA and COMIT for SAT. Moreover, the superiority of elitist algorithms over sMMA (mid-symbol in each entry of Table 2) is also statistically significant in most cases. Among the different elitist EDAs, COMIT_e seems to provide the best overall results, being the top algorithm in TRAP (see Fig. 1 for an illustration of fitness evolution on this problem) and HXOR, and being indistinguishable from PBIL_e in SAT. It is interesting to notice the good performance of UMDA_e on this problem. This can be due to the fact that the optimal solution in this case is a homogeneous string (all 0s or all 1s), a structure which is easily captured by memes; as soon as the simpler nature of UMDA’s probabilistic model directs the search towards a state with predominance of either symbol, the memes can facilitate reaching the optimum. This hypothesis is supported by the comparatively worse results of univariate algorithms on the HXOR problem, in which the optimal solution contains a 50%-50% mixture of 1s and 0s, placed in precise locations (so as to make any half of the solution be maximally dissimilar from the other half).

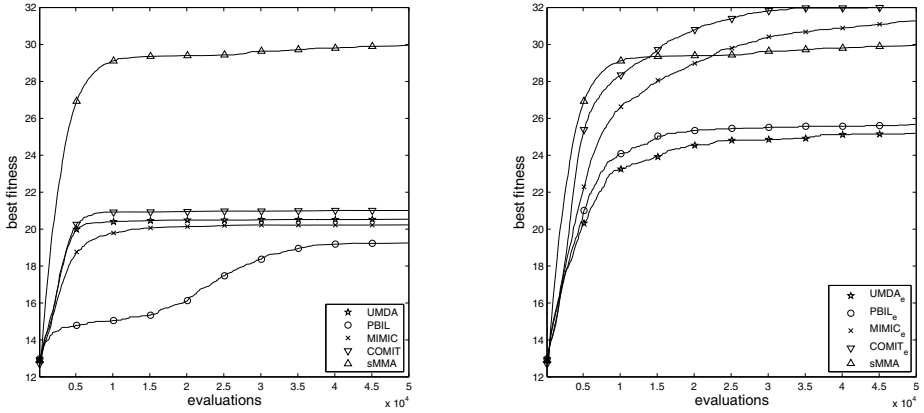


Fig. 1. Evolution of best fitness in TRAP for non-elitist multimemetic EDAs (left) and elitist ones (right). The results of sMMA are included in both figures.

Table 3. Results (20 runs) of the different MMAs on TRAP, HIFF, HXOR and SAT, without Laplace correction in meme probability estimation. The median (\tilde{x}), mean (\bar{x}) and standard error of the mean ($\sigma_{\bar{x}}$) are shown.

	TRAP		HIFF		HXOR		SAT	
	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$
UMDA _e *	23.8	25.2 ± 0.8	576.0	516.9 ± 17.3	342.0	342.8 ± 7.6	549.0	548.4 ± 0.3
PBIL _e *	24.7	25.7 ± 0.7	443.0	451.5 ± 15.9	262.5	259.9 ± 2.2	549.0	548.6 ± 0.3
MIMIC _e *	32.0	31.7 ± 0.2	464.0	473.6 ± 12.6	412.0	417.0 ± 5.3	548.5	548.5 ± 0.2
COMIT _e *	32.0	32.0 ± 0.0	464.0	480.0 ± 13.9	416.0	427.6 ± 10.0	548.0	548.4 ± 0.3

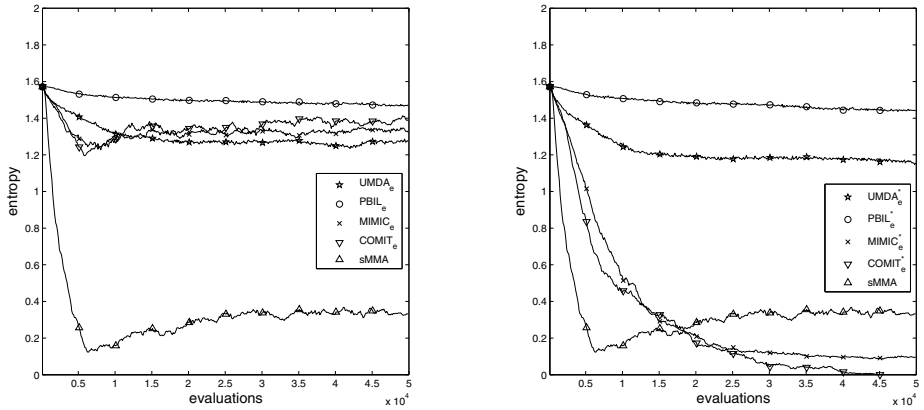


Fig. 2. Evolution of meme diversity in TRAP for multimemetic EDAs using Laplace correction in the probabilistic modeling of memes (left) and without using such correction (right)

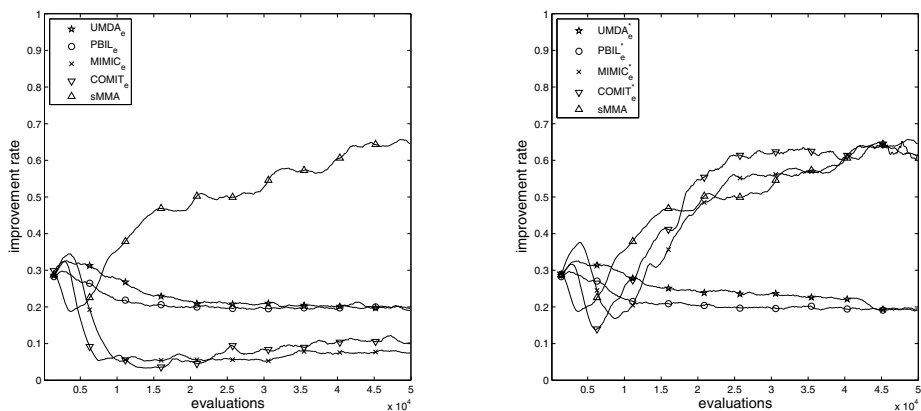


Fig. 3. Evolution of meme success (percentage of meme applications that result in an improvement) in TRAP for multimemetic EDAs using Laplace correction in the probabilistic modeling of memes (left) and without using such correction (right)

Let us now turn our attention to the effect that the utilization of Laplace correction has on the algorithms. While its use is fundamental within the genotypic model (experiments without this correction indicate quick convergence of the probabilistic model to suboptimal states within a few dozens generations), its use at the memetic level seems more questionable. As it can be seen in Fig. 2 (left), the entropy of memes generated remains very high in all EDAs, unlike the sMMA which stabilizes at a low entropy level. This indicates that the EDAs are facing difficulties to focus the search of adequate memes, either by the limitations of the underlying probabilistic model or by the exploratory disturbance of the Laplace correction. We have therefore performed experiments with the elitist EDAs deactivating this correction in meme modeling. Not surprisingly, this has a larger influence in the bivariate models, which are now capable of converging to particular states, than in univariate models, which remain incapable of grasping the structure of memes in many cases – see Fig. 2 (right). This also has in general a positive influence in performance as shown in Table 3. While the univariate models perform slightly better, the difference is not statistically significant. In the case of the bivariate models, there is a statistically significant difference in favor of MIMIC_e^{*} (the superscript ^{*} denoting deactivation of Laplace correction) for TRAP and HXOR, and in favor of COMIT_e^{*} for HIFF. It is interesting to notice how meme success (the percentage of meme applications that result in an improvement) is higher in these variants – see Fig. 3 – supporting the hypothesis that the search is more focused in this case.

4 Conclusions

We have studied the use of EDAs in a multimemetic context. They appear to be a promising approach to this kind of self-adaptive memetic optimization due to their

well-known advantages, namely their requiring less parameterization effort than their genetic counterparts and their amenability to model combinatorial structures such as memes. Indeed, the experimentation with multimemetic EDAs has provided encouraging results: when endowed with elitism, multimemetic EDAs are markedly superior to a MMA manipulating genes and memes using genetic operators. We have also observed that the memetic search is more focused when no Laplace correction is used in meme modeling. Of course, this may need further investigation in other contexts in which memes are represented in a different way (and indeed in the general context of EDA optimization, since the use of this technique is not widespread). This is not the only line for future research: on one hand, the use of more complex probabilistic graphical models such as Bayesian networks is an appealing option. As a matter of fact, it may be conceivable that the structure used to model the probability distribution over the memetic space be different than its genotypic counterpart. This could pave the way to a fully decoupled evolutionary model in which genotypes and memes evolve in different populations, subject to separate selection processes, and interacting via some strategy for genotype-meme pairing and application, in the line of coevolutionary memetic algorithms [20].

Acknowledgements. This work is partially supported by MICINN project ANYSELF (TIN2011-28627-C04-01), by Junta de Andalucía project DNEMESIS (P10-TIC-6083) and by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

References

1. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: 14th International Conference on Machine Learning, pp. 30–38. Morgan Kaufmann (1997)
2. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. CMU-CS-95-19, Carnegie Mellon University, Pittsburgh, PA, USA (1994)
3. Bonet, J.S.D., Isbell, C.L., Jr., Viola, P.: Mimic: Finding optima by estimating probability densities. In: Mozer, M., Jordan, M., Petsche, T. (eds.) *Advances in Neural Information Processing Systems*. vol. 9, pp. 424–430. The MIT Press (1996)
4. Cestnik, B.: Estimating probabilities: A crucial task in machine learning. In: Aiello, L. (ed.) 9th European Conference on Artificial Intelligence, Pitman, Stockholm, Sweden, pp. 147–149 (1990)
5. Dawkins, R.: *The Selfish Gene*. Clarendon Press, Oxford (1976)
6. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) *Second Workshop on Foundations of Genetic Algorithms*, pp. 93–108. Morgan Kaufmann, Vail (1993)
7. González, C., Lozano, J., Larrañaga, P.: Mathematical modeling of discrete estimation of distribution algorithms. In: Larrañaga, P., Lozano, J.A. (eds.) *Estimation of Distribution Algorithms, Genetic Algorithms and Evolutionary Computation*, vol. 2, pp. 147–163. Springer, US (2002)

8. Hart, W., Krasnogor, N., Smith, J.: Memetic Evolutionary Algorithms. In: Hart, W.E., Smith, J.E., Krasnogor, N. (eds.) *Recent Advances in Memetic Algorithms*. STUDFUZZ, vol. 166, pp. 3–27. Springer, Berlin (2005)
9. Krasnogor, N., Blackburne, B., Burke, E., Hirst, J.: Multimeme algorithms for protein structure prediction. 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. 769–778. Springer, Heidelberg (2002)
10. Larrañaga, P., Lozano, J.A.: *Estimation of Distribution Algorithms. Genetic Algorithms and Evolutionary Computation*, vol. 2. Springer, US (2002)
11. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E.: *Towards a New Evolutionary Computation*. STUDFUZZ, vol. 192. Springer, Heidelberg (2006)
12. Moscato, P.: *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*. Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA (1989)
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. Neri, F., Cotta, C., Moscato, P.: *Handbook of Memetic Algorithms*. SCI, vol. 379. Springer, Heidelberg (2012)
15. Noguera, R., Cotta, C.: Analyzing meme propagation in multimemetic algorithms: Initial investigations. In: 2013 Federated Conference on Computer Science and Information Systems, pp. 1013–1019. IEEE Press, Cracow (2013)
16. Pelikan, M., Goldberg, D.: Hierarchical BOA solves ising spin glasses and MAXSAT. In: Cantú-Paz, E., et al. (eds.) *GECCO 2003*. LNCS, vol. 2724, pp. 1271–1282. Springer, Heidelberg (2003)
17. Pelikan, M., Sastry, K., Cantú-Paz, E.: Scalable Optimization via Probabilistic Modeling. SCI, vol. 33. Springer, Heidelberg (2006)
18. Radetic, E., Pelikan, M., Goldberg, D.E.: Effects of a deterministic hill climber on hBOA. In: 2009 Genetic and Evolutionary Computation Conference, pp. 437–444. ACM, New York (2009)
19. Robles, V., Miguel, P., Larrañaga, P.: Solving the traveling salesman problem with EDAs. In: Larrañaga, P., Lozano, J.A. (eds.) *Estimation of Distribution Algorithms, Genetic Algorithms and Evolutionary Computation*, vol. 2, pp. 211–229. Springer, US (2002)
20. Smith, J.E.: Self-adaptative and coevolving memetic algorithms. In: Neri, F., Cotta, C., Moscato, P. (eds.) *Handbook of Memetic Algorithms*. SCI, vol. 379, pp. 167–188. Springer, Heidelberg (2012)
21. Watson, R.A., Pollack, J.B.: Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In: 1999 IEEE Congress on Evolutionary Computation, pp. 292–297. IEEE Press, Washington D.C. (1999)
22. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem. In: Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.) *Towards a New Evolutionary Computation*. STUDFUZZ, vol. 192, pp. 281–292. Springer, Heidelberg (2006)

Factoradic Representation for Permutation Optimisation

Olivier Regnier-Coudert and John McCall

IDEAS Research Institute, Robert Gordon University, Aberdeen, UK
{o.regnier-coudert,j.mccall}@rgu.ac.uk

Abstract. It is known that different classes of permutation problems are more easily solved by selecting a suitable representation. In particular, permutation representations suitable for Estimation of Distribution algorithms (EDAs) are known to present several challenges. Therefore, it is of interest to investigate novel representations and their properties. In this paper, we present a study of the factoradic representation which offers new modelling insights through the use of three algorithmic frameworks, a Genetic Algorithm (GA) and two EDAs. Four classic permutation benchmark problems are used to evaluate the factoradic-based algorithms in comparison with published work with other representations. Our experiments demonstrate that the factoradic representation is a competitive approach to apply to permutation problems. EDAs and more specifically, univariate EDAs show the most robust performance on the benchmarks studied. The factoradic representation also leads to better performance than adaptations of EDAs for continuous spaces, overall similar performance to integer-based EDAs and occasionally matches results of specialised EDAs, justifying further study.

Keywords: Estimation of Distribution Algorithms, Factoradics, Permutation.

1 Introduction

The permutation representation is widely used to model solutions to optimisation problems. Although it is often seen as a natural way to represent solutions, it also appears to be a challenging domain to model because of alleles being interconnected. Recent work has highlighted this challenge for Estimation of Distribution Algorithms (EDAs) and proposed solutions that model the space of permutations by means of specialised distribution models [1].

A different approach to overcome challenges encountered when handling permutations is to introduce alternative genotypes. In Evolutionary Algorithms (EAs), the term *genotype* is often used to describe the domain searched by the algorithms, that is the search space on which operators are applied. In order to assess solutions, a phenotype is required. The *phenotype* represents the domain in which a solution can be evaluated, or in other words, a domain that can be read by the fitness function. Not only does using alternative genotypes allow

some problems to be modelled efficiently by EAs, but it may also map a problem to a domain which is more adapted to these algorithms. Consequently, it has been shown that using many representations within the same search procedure may yield improved results by balancing out between the biases introduced by each representation [2].

With respect to permutations, the random key (RK) genotype has widely been used [3]. Yet, it is known to display some features that may inhibit the search in some contexts [4]. There also exist in the literature mentions of an alternative genotype for permutations based on the factorial numbering system, also referred to as factoradics [5]. Despite interesting features, the factoradic genotype has been little studied by the EA community. The present paper proposes directions to using the factoradic representation for optimisation. The paper aims to understand whether the factoradic representation can support search in EAs and identify the contexts in which it is more likely to do so, through the investigation of different specialised operators and algorithmic frameworks including a Genetic Algorithm (GA), and two distinct EDAs.

2 Factoradic Representation

The factoradic system is a numbering system of dimension n , which uniquely represents each number between 0 and $n! - 1$ as a string of factoradic digits. Each position i , $i \in [0, n - 1]$ can be assigned a digit taking a value between 0 and i . The base of each position increases with i and so does its place value, i.e. the size of the factorial. Thus, the place value at position i is $i!$. The factoradic $a_{(1)}$ can be transformed into its decimal form $a_{(10)}$ as follows:

$$a_{(10)} = \sum_{i=0}^{n-1} a_{(1)_i} \times i!, \quad (1)$$

where $a_{(1)_i}$ represents the i -th element of $a_{(1)}$. The potential of factoradics goes beyond the simple numbering system as it represents a way to easily represent permutations. For example, the factoradic 422100 denotes the permutation where the 4th, 2nd, 2nd, 1st, 0th and 0th items are drawn successively without replacement from the set of items. Figure 1 illustrates how this factoradic number represents the permutation 423105 .

The factoradic representation allows representation of a permutation by a string of integers of similar size, in which each digit is a number in $[0, i]$. In addition, it introduces different weights between positions. These characteristics of factoradics allow a straightforward application of EA and more precisely EDA techniques in the permutation domain without losing problem properties. To our knowledge, most of the applications of factoradics in EAs have focused on Particle Swarm Optimisation (PSO) to turn permutations into a usable form for the algorithms [6]. Factoradics have also proved useful in allowing restriction of the search to sub spaces [5]. We refer the reader to the latter study for a more detailed description of factoradics.

Remaining indices	Factoradic	Permutation
0 1 2 3 4 5	4	4
0 1 2 3 5	4 2	4 2
0 1 3 5	4 2 2	4 2 3
0 1 5	4 2 2 1	4 2 3 1
0 5	4 2 2 1 0	4 2 3 1 0
5	4 2 2 1 0 0	4 2 3 1 0 5

Fig. 1. Mapping from factoradic to permutation

We consider simple operators for the factoradic representation. We note first that standard single-point, multi-point and uniform crossover can be applied to factoradics without alteration. Mutation operators require more care because of the variable limit on the i th digit. We present three mutation operators suitable for the factoradic representation. First, the point mutation (PM) only affects one allele of the mutated solution. However, because of the characteristics of the factoradic representation, the position of the allele influences the amount of disruption to the solution. Hence, PM needs to be defined in conjunction with a mutation distance as defined in [7] for permutations. The mutation distance d denotes the position of the gene to mutate. Note that the gene at position zero can only take the zero value and is thus never considered during operations. An allele is mutated by sampling randomly its value from the range $[0, d]$. The second mutation defined for the factoradic domain is the multi-point mutation (MPM), which performs a PM on all alleles at a position of similar or lower value than the specified mutation distance. Consequently, MPM is expected to be more disruptive than the simple PM. Finally and in order to offer an even more disruptive operator, the random multi-point mutation (RMPM) is introduced. RMPM selects at random d alleles to mutate, regardless of their order.

3 Factoradic Algorithms

Factoradic Genetic Algorithm. The basic concept of the GA developed for the experiments is presented in Algorithm 1. ϵ denotes the size of the elitism, while crossover and mutation rates are referred to as α and β . Starting from an initial randomly generated population pop , the GA copies the best solutions according to ϵ , select solutions par_1 and par_2 for recombination and performs successively crossover and mutation with respect to α and β . Generated solutions are added to the new population pop_{new} until it reaches the population size, in which case it replaces the old population before being re-evaluated.

Factoradic Univariate Estimation of Distribution Algorithm. An EDA is based on the concept of evaluating a population of solutions and building a model from a selected subset of this population. This model can then be used to sample new solutions. In a univariate EDA, we construct a fully factorised

Algorithm 1. *GA*

```

Generate and evaluate pop
for each generation g do
  popnew = ∅
  Add  $\epsilon$  best solutions to popnew
  repeat
    Select parents par1 and par2
    Generate offspring off by applying crossover to par1 and par2, with probability
     $\alpha$  or by copying par1 with probability  $(1 - \alpha)$ 
    Apply mutation to off with probability  $\beta$ 
    Add off to popnew
  until ( $|pop_{new}| = |pop|$ )
  pop = popnew
  Evaluate pop
end for
Return pop

```

probabilistic model which is sampled as a set of independent marginals. Algorithm 2 describes the univariate EDA used in the present study, based on the Population-Based Incremental Learning algorithm (PBIL) [8], but applied to the factoradic representation. First, the model $\mathcal{M}(i, j)$ is initialised with uniform probabilities. $\mathcal{M}(i, j)$ essentially gathers the marginal probability for each item j to be in position i . A population *pop* is generated at random and evaluated. At each generation g , a subset of *pop*, *pop_{sel}* is selected and the model is updated. This is done using the relative frequency of each item j at each position i in *pop_{sel}*. Note that the notation *pop_{sel}*(k) is used to denote the k th solution of *pop_{sel}*. A model $\mathcal{M}_{temp}(i, j)$ is first created, considering only the frequencies obtained from the current population. This model is then used to update the previous model $\mathcal{M}(i, j)$. The learning rate γ defines how conservative the update is. A high γ results in the model being mostly based on the current population, while a low γ implies that the model keeps a lot of features from the previous generation. Note that setting γ to 1 results in the algorithm to be the Univariate Marginal Distribution Algorithm (UMDA) [9], where the model used at each generation is only built from the information obtained from the current population. Once updated, the model is sampled to generate the new population and the process repeated over several generations.

Factoradic COMpetitive Mutating Agents. The COMMA framework [7] evolves a population of agents. Each agent is assigned a solution whose fitness is used to rank agents within the population. COMMA generates a distribution of solutions spaced within a disruption distance of each agent's solutions. This distribution is geometrically sampled by means of mutation operators to produce a new solution for each agent. As illustrated in Algorithm 3 for minimisation optimisation, the principle of COMMA is to apply different operators to the agents according to their rank in order to perform both exploration and exploitation of

Algorithm 2. *PBIL*

```

Initialize model  $\mathcal{M}(i, j)$  with uniform probabilities
Generate and evaluate  $pop$ 
for each generation  $g$  do
  Select  $pop_{sel}$  from  $pop$ 
  for each index  $i, i < n$  do
    for each item  $j, j < i$  do
       $\mathcal{M}_{temp}(i, j) = \frac{\sum_{k=0}^{|pop_{sel}|} x_i}{|pop_{sel}|}$ , with  $\begin{cases} x_i = 1, & \text{if } pop_{sel}(k) = j \\ x_i = 0, & \text{otherwise} \end{cases}$ 
       $\mathcal{M}(i, j) = \gamma \mathcal{M}_{temp}(i, j) + (1 - \gamma) \mathcal{M}(i, j)$ 
    end for
  end for
   $pop_{new} = \emptyset$ 
  repeat
    Sample solution from  $\mathcal{M}(i, j)$  and add to  $pop_{new}$ 
  until ( $|pop_{new}| = |pop|$ )
   $pop = pop_{new}$ 
  Evaluate  $pop$ 
end for

```

the search space. Applying such operators allows to sample new solutions more or less distant in the search space from the agents' solutions. The combinations of solutions and operators represent the model in COMMA. In the case of the factoradic implementation, the operators defined in Section 2 are adapted because they present a notion of mutation distance. COMMA operates as follows. For each position pos_j in the population pop sorted in ascending order, a mutation distance d_j is set such that for two agents at positions e and f , $d_e \leq d_f$ if $e < f$. Each agent a_i is initially assigned a random solution s_i . The population is then sorted by fitness. At each generation, each agent mutates s_i using the distance $dist_i \in [1, d_r]$ defined according to its position r in the population. Note that if the boolean parameter *fixedDistance* is true, $dist_i = d_r$. If the mutated solution s_{new} has a better fitness than s_i , a_i replaces s_i with s_{new} .

4 Experiments

4.1 Test Problems

Travelling Salesman Problem. Based on a given set of k cities and a matrix of distances d_{ij} between all pairs of cities $\{i, j\}$, the TSP aims to determine a shortest possible route r that visits each city exactly once. The route may start at any city, but should end where it started. Hence r is a vector of length k . Formally and using r_a to denote the a -th city in r , the TSP is expressed as:

$$\min\left\{\left(\sum_{a=0}^{k-1} d_{r_a, r_{a+1}}\right) + d_{r_k, r_0}\right\} \quad (2)$$

Algorithm 3. *COMMA* (for minimisation)

```

Initialize pop of  $\sigma$  agents with random solutions, distance vector  $d$  of size  $\sigma$ 
repeat
  Sort pop by fitness in descending order
  for each agent  $a_i, i \in [0, \sigma - 1]$  do
    Get position  $r$  of  $a_i$  in pop
    if fixedDistance then
       $dist_i = d_r$ 
    else
      Select  $dist_i$  with uniform probability from  $[1, d_r]$ 
    end if
    Sample new solution  $s_{new}$  with fitness  $fit_{new}$  by mutating  $s_i$  with distance  $dist_i$ 

    if  $fit_{new} < fit_i$  then
      Assign  $s_i = s_{new}$ 
    end if
  end for
until Stopping condition met

```

Permutation Flowshop Scheduling Problem. In the PFSP [10], a set of jobs is given that need to be run on a set of machines. Each of the jobs has to be processed on every machine exactly once and only one job can be handled by a given machine at a given time. It is assumed that each job is processed on the machines in a set order and that the time required to process jobs varies between jobs and between machines. The objective of the PFSP is to minimize the time of completion of the last submitted job on the last machine. The general expression to calculate the time of completion c_{π_i} of a given job π_i on a machine j , given its corresponding processing time $t_{\pi_i,j}$ is given in (3). The fitness of a sequence of jobs represented as the permutation π can be derived as in (4), where n and m respectively stand for the total number of jobs and machines.

$$c_{\pi_i,j} = \max\{c_{\pi_{i-1},j}, c_{\pi_i,j-1}\} + t_{\pi_i,j} \tag{3}$$

$$c_\pi = c_{\pi_n,m} \tag{4}$$

Quadratic Assignment Problem. In QAP [11], n facilities are to be assigned to n locations in such way that the total amount of resources being transferred between locations is minimized. Flows $f_{a,b}$ between all pairs of facilities (a, b) and distances $d_{l(a),l(b)}$ between all pairs of locations $l(a), l(b)$ are known. Mathematically the problem can be formulated as (5).

$$\min\left\{\sum_{a,b} f_{a,b}d_{l(a),l(b)}\right\} \tag{5}$$

Linear Ordering Problem. The aim of the LOP is to find a simultaneous permutation ω of the rows and columns of a matrix $D = (d_{ij})$ that maximizes the sum of the superdiagonal entries. Formally, its objective is defined as follows.

$$\max\left\{\sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{\omega_i, \omega_j}\right\} \quad (6)$$

4.2 Experimental Settings

Because of the wide range of EDAs for permutation problems considered, [12] was chosen as the basis for comparison. A similar limit on fitness evaluations was set, that is $1000n^2$ for each problem of size n and 10 runs were needed to compute each result set. In order to reduce the importance given to parameter setting, default parameter values were initially chosen and the algorithms run with every possible combination. Default values are given in Table 1. Note that crossover and mutation rates were respectively set to 0.9 and 0.1. Algorithm performance was measured by best fitness found at the end of the run and by computing the relative percentage deviation (RPD) to the known optimum, as described in [1]. For all comparisons, statistical significance (95% confidence interval) was measured by means of unpaired t-test, applying Bonferroni correction.

Table 1. Default parameter values

Parameter	Default values	Parameter	Default values
pop size	50, 100, 500, 1000	tournament size	0.1, 0.25, 0.5
elitism	0, 1	selection ratio	0.1, 0.25, 0.5, 0.75
mutation	PM, MPM, RMPM	learning rate	0.5, 0.7, 0.9, 1
crossover	1-point, 2-point	fixed distance	true, false

5 Results and Discussion

Suitability of Frameworks for Factoradics. Figure 2 shows the RPD of all methods on each problem. Over all instances, COMMA and PBIL show the best performance, with the exception of TSP. On PFSP and QAP, COMMA and PBIL are the most robust methods, although COMMA presents smaller standard deviations than PBIL. LOP is the problem on which the biggest difference in performance is observed. While COMMA shows relatively poor results, PBIL is significantly better than the other methods. Overall, PBIL appears as a good compromise to handle the factoradic representation across problems. It was also observed that setting high learning rates brought enhanced results. Consequently and as can be seen in Table 2, UMDA often outperforms PBIL.

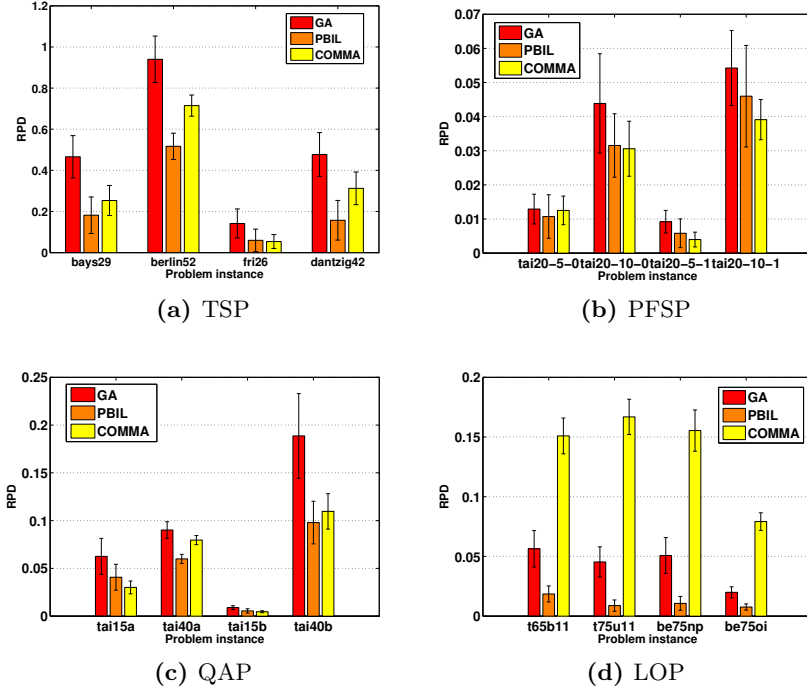


Fig. 2. RPD of the factoradic frameworks on the four problems

Suitability of Factoradics for the Selected Problems. To evaluate the suitability of the factoradic representation for permutation optimisation, the results of the best algorithm with the most efficient settings is compared with the suite of methods investigated in [12] for each problem. Results are presented in Table 2, showing the best factoradic framework and the fitness of the best obtained solution. Table 2 also shows for each problem the EDAs that outperform the factoradic methods and the EDAs that match their results. Overall, the performance obtained using factoradics is close to the one of the best integer-based EDAs used for permutation optimisation, that is UMDA, MIMIC and $EBNA_{BIC}$, described as providing good solutions. Continuous EDAs, such as $UMDA_c$ and $EGNA_{ee}$, generally show poor performance on permutation problems. As one might expect of a more natural representation, the factoradic implementations exhibit solutions of greater quality. Experiments from [12] showed that specialised EDAs are the most efficient and more precisely the EDAs using edge and node histogram models, EHBSA and NHBSA. Although the comparison with these algorithms shows that factoradic methods do not always match their results, there exist problems where performances are of the same magnitude. Also note that the IDEA-ICE permutation-based EDA never exhibits better outcome than the factoradic frameworks.

Finally, the recursive EDA (REDA) and OmeGA, a GA based on the RK representation, are generally behind the proposed methods. Direct comparison between the factoradic GA and OmeGA shows that the two algorithms perform at the same level on four instances, mostly on PFSP. However, the factoradic GA outperforms OmeGA to a significant extent on all LOP and TSP instances and most of the QAP ones. This comparison highlights the advantage of using factoradics over RK in a GA.

Table 2. Results from unpaired t-tests. A \checkmark symbol denotes the algorithms whose performance is not significantly different from the best factoradic implementation on the problem. \blacksquare shows algorithms whose results outperform those of the factoradic algorithms. Empty cells show methods that are outperformed by implementations using the factoradic representation.

Problem	Best Algo.	Best Solution	UMDA	MIMIC	EBNABIC	TREE	UMDA _c	EGNA _{ec}	IDEA – ICE	EHBSA _{WT}	EHBSA _{WO}	NHBSA _{WT}	NHBSA _{WO}	REDA _{UMDA}	REDA _{MIMIC}	OmeGA
TSP-bays29	PBIL	2387.4 (179.5)	\blacksquare	\checkmark	\checkmark					\blacksquare	\blacksquare	\blacksquare	\blacksquare			
TSP-berlin52	PBIL	11440.7 (481.6)	\blacksquare	\checkmark	\blacksquare					\blacksquare	\blacksquare	\checkmark	\blacksquare			
TSP-fri26	COMMA	982.3 (38.8)	\checkmark		\checkmark		\checkmark	\checkmark		\blacksquare	\blacksquare	\checkmark	\blacksquare			
TSP-dantzig42	PBIL	805.3 (72.9)	\checkmark		\checkmark		\blacksquare	\blacksquare	\checkmark	\blacksquare	\blacksquare	\checkmark	\blacksquare			\blacksquare
PFSP-tai20-5-0	PBIL	1291.7 (8.2)	\checkmark	\checkmark	\checkmark				\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark		\checkmark
PFSP-tai20-10-0	COMMA	1630.4 (12.8)	\checkmark	\checkmark	\checkmark					\blacksquare	\checkmark	\blacksquare	\blacksquare			
PFSP-tai20-5-1	COMMA	1364.4 (2.9)	\checkmark	\checkmark		\checkmark			\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark		\checkmark
PFSP-tai20-10-1	COMMA	1723.9 (9.8)	\checkmark	\checkmark	\checkmark					\blacksquare	\checkmark	\blacksquare	\blacksquare			
QAP-tai15a	COMMA	399889 (2610)	\checkmark	\checkmark	\checkmark	\checkmark				\checkmark		\blacksquare	\checkmark			
QAP-tai40a	PBIL	3327464 (14966)	\blacksquare	\blacksquare	\blacksquare							\checkmark	\blacksquare			
QAP-tai15b	COMMA	52002721 (54819)	\checkmark	\checkmark	\checkmark	\checkmark				\checkmark		\blacksquare	\blacksquare			
QAP-tai40b	UMDA	699677162 (14205322)	\checkmark	\checkmark	\blacksquare	\checkmark				\blacksquare	\checkmark	\blacksquare	\checkmark			
LOP-t65b11	UMDA	350134 (2379)	\checkmark	\checkmark	\checkmark					\blacksquare		\blacksquare	\blacksquare			
LOP-be75np	UMDA	709328 (4182)	\checkmark	\checkmark	\checkmark					\blacksquare		\blacksquare	\blacksquare			
LOP-be75oi	PBIL	110323 (287)	\checkmark	\checkmark	\checkmark					\blacksquare		\blacksquare	\blacksquare			

6 Conclusions

In this paper, we have presented the factoradic representation as a genotype for permutations, along with frameworks that can be employed to make use of it. Experiments on benchmark problems have shown that the factoradic representation is suitable for permutation optimisation, especially when used within algorithms such as univariate EDAs. Comparison with other studies has demonstrated that algorithms using factoradics present matching performance with integer-based EDAs and can compete with some specialised EDAs. Future work should focus on building a deeper understanding of the relation between factoradics and

problem characteristics. Alternative ways to model factoradics such as tree-based approaches could also be explored. Finally, given the promising results of PBIL, the development of multivariate factoradic-based EDAs represents an interesting avenue for further research.

References

1. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Transactions on Evolutionary Computation* (2013)
2. Schnier, T., Yao, X.: Using multiple representations in evolutionary algorithms. In: *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, pp. 479–486. IEEE (2000)
3. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing* 6(2), 154–160 (1994)
4. Ashlock, D.: *Evolutionary computation for modeling and optimization*. Springer (2006)
5. Mehdi, M.: *Parallel hybrid optimization methods for permutation based problems*. PhD thesis, Université des Sciences et Technologie de Lille (2011)
6. Samarghandi, H., ElMekkawy, T.Y.: A meta-heuristic approach for solving the no-wait flow-shop problem. *International Journal of Production Research* 50(24), 7313–7326 (2012)
7. Regnier-Coudert, O., McCall, J., Ayodele, M.: Geometric-based sampling for permutation optimization. In: *Proceeding of the 2013 Annual Conference on Genetic and Evolutionary Computation Conference*, pp. 399–406. ACM (2013)
8. Baluja, S.: *Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning*. Technical report, Carnegie Mellon University (1994)
9. Mühlenbein, H.: The equation for response to selection and its use for prediction. *Evolutionary Computation* 5(3), 303–346 (1997)
10. Ruiz, R., Maroto, C.: A comprehensive review and evaluation of permutation flow-shop heuristics. *European Journal of Operational Research* 165(2), 479–494 (2005)
11. Lawler, E.L.: The quadratic assignment problem. *Management science* 9(4), 586–599 (1963)
12. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence* 1, 103–117 (2012)

Combining Model-Based EAs for Mixed-Integer Problems

Krzysztof L. Sadowski¹, Dirk Thierens¹, and Peter A.N. Bosman²

¹ Utrecht University, The Netherlands

² CWI Amsterdam, The Netherlands

Abstract. A key characteristic of Mixed-Integer (MI) problems is the presence of both continuous and discrete problem variables. These variables can interact in various ways, resulting in challenging optimization problems. In this paper, we study the design of an algorithm that combines the strengths of LTGA and iAMaLGaM: state-of-the-art model-building EAs designed for discrete and continuous search spaces, respectively. We examine and discuss issues which emerge when trying to integrate those two algorithms into the MI setting. Our considerations lead to a design of a new algorithm for solving MI problems, which we motivate and compare with alternative approaches.

1 Introduction

Mixed-Integer (MI) optimization problems arise in many real-world application domains. A key characteristic of MI problems is the presence of both continuous and discrete problem variables. Many studies exist on dealing with either continuous or discrete search spaces only. We are interested in studying if and how approaches originally designed for real or discrete domains only can be integrated for the mixed-integer landscapes.

More specifically, we consider two state-of-the-art model building EAs: LTGA [7] and iAMaLGaM [1]. Both were previously shown to exhibit excellent polynomial scale-up behavior on various well-known black-box benchmark problems. We wish to study if making use of the model building and learning abilities of both these algorithms can be applied to MI problems while retaining some of the excellent scale-up behavior. The model-building nature of these algorithms allows us to consider black-box problems where no prior information about a problem structure is known. Some research on solving MI problems with EDAs has been discussed in [4], but was limited in terms of possible variable dependencies.

We introduce an integrated implementation which relies on interleaving the model-building capabilities of both EAs. Integrative dependency processing is the holy grail of an approach using model-building algorithms to solve MI problems. It is important to first understand the capacity and limitations of an approach that interleaves existing individual models, as it allows us to better understand the requirements for integrative dependency processing. A crucial aspect of designing such algorithm is determining a way of maintaining a proper balance between structure learning and offspring creation done by the two independent models. Obtaining such balance is a difficult task, as many

challenges arise when dealing with MI landscapes, which do not exist in only discrete or continuous spaces. Because of the nature of MI landscapes, performing independent learning with different models for the continuous and discrete variables can easily lead to premature convergence when some variables are not sufficiently explored, or to over-exploration when too much focus is given to some variables. How difficult is it to achieve a proper evaluation balance and adequate scalability as the problem size increases? Is it even possible to solve dependent problems where continuous variables interact with the discrete ones, while using integrated but independently learning models?

In order to answer these and other questions, we design and study mixed-integer landscapes with different levels of variable interactions: no interactions, binary and/or continuous interactions only, and interactions between both types of variables. We identify various problematic issues, and design methodologies to counteract those issues. We also examine the performance of the existing Mixed-Integer Evolution Strategy (MIES) [3] [5] on our benchmark set.

2 Background

Our approach to solving Mixed-Integer problems focuses on bringing together two model-building algorithms, LTGA and iAMaLGaM, and carefully integrating them.

2.1 LTGA

The Linkage Tree Genetic Algorithm (LTGA) is a state-of-the-art model building GA designed for solving discrete problems [7] [8]. It makes use of a hierarchical clustering algorithm in each generation in order to learn variable dependencies, which are represented via a linkage tree. In this model, each node of the linkage tree is a subset between one and $l_d - 1$ problem variables which form an important building block for the solution.

During each generation, LTGA iterates over all solutions in the population in an attempt to improve them: For each solution the linkage tree is traversed and each subset of the tree is used as a crossover mask between a donor and the parent solution. A donor is selected randomly from the population for each subset mask. In other words, the values of variables clustered together at a given node of the linkage tree are copied from a donor onto the parent solution. A result of each such crossover is immediately evaluated. If the resulting offspring solution is better or equal than its parent, it instantly replaces the parent. Otherwise, the offspring is discarded. This process is repeated until all the linkage tree nodes are processed. This algorithm has been shown to work very efficiently for various discrete problems [8].

2.2 iAMaLGaM

Incremental Adapted Maximum-Likelihood Gaussian Model Iterated Density Estimation Evolutionary Algorithm (iAMaLGaM) is a state-of-the-art EDA for real-valued black-box optimization (BBO) [1]. Following the general EDA paradigm, iAMaLGaM estimates a probability distribution every generation from the selected solutions and

generates new solutions by sampling the estimated distribution. The probability distribution used in iAMaLGaM is the Gaussian distribution. The mean vector and covariance matrix are estimated incrementally using memory decay on maximum-likelihood estimates. Risk of premature convergence is counteracted by a mechanism which scales up the co-variance matrix when needed. Finally, Anticipated Mean Shift procedure is implemented to improve the algorithm behavior in slope-like regions of the search space. All these factors contribute to iAMaLGaM achieving very good scale-up and rotation-invariant behavior on many well known BBO benchmarks [2].

3 Integrated Algorithm

A solution to a mixed problem with a total problem length $l = l_d + l_c$, where l_d and l_c are the number of discrete and continuous variables respectively, is of the form:

$$X = X_d X_c = d_0 \dots d_{l_d-1} c_0 \dots c_{l_c-1}$$

where $d_i \in \{0, 1\}$, $c_i \in \mathbb{R}$ and X_d , X_c are the sets of all discrete and continuous variables, respectively.

Our integrated algorithm builds models over the two subspaces independently. Models can be build and exploited in various ways. A balance between the rates at which this happens is likely to play an important role in the convergence properties of the algorithm. The models in question have some common properties, which we can use as a backbone for integration. Both models attempt to improve and generate new offspring for each solution in the population during one generation, and learn a new model at the beginning of a next generation. However, after a new model is generated, iAMaLGaM creates the new continuous solutions by sampling from the new model. Once all the offspring solutions are created, the generation ends. This means that for a population of size n , iAMaLGaM performs n function evaluations within one generation. This differs from the generational procedure of LTGA. The linkage tree contains $2l_d - 1$ nodes. Following the variation procedure of LTGA, a solution may need to be evaluated $2l_d - 1$ times, giving the upper bound of $n * (2l_d - 1)$ evaluations overall during one entire generation. A straightforward approach of directly merging these models by keeping their generations synchronized might not work well, as depending on the ratio of discrete to continuous variables in a MI setting, one model could dominate the other by exploring some areas of the problem too heavily, while leaving other regions potentially not explored enough. This could lead to premature convergence or unnecessary over-exploration of the search space. To address this potential imbalance, we introduce the integrated EA in Figure 1. In this algorithm, the generational progress of the different models is not the same, and takes into account the proportions between the number of discrete and continuous problem variables. The continuous model is re-learned after every solution in the population has been sampled. The discrete model however is only re-learned after all of the $2l_d - 1$ nodes of the linkage tree have been processed for all solutions. This balances the discrete and continuous evaluations much better regardless of the ratio between those types of variables.

The algorithm generates the initial population randomly. Each solution \mathcal{P}_i in the population consists of a continuous component \mathcal{P}_{i_c} as well as a discrete component \mathcal{P}_{i_d} . \mathcal{X}_i

is the offspring solution. The core of the algorithm has two nested loops, which iterate over each linkage tree subset, and additionally iterate over every solution for each of the subsets. The continuous model is learned after a given subset was applied to each solution. The discrete model, however, is learned only after all the subsets have been tried on all solutions. This process continues until a termination condition is reached. New solution acceptance criteria also differ. The continuous model is learned from the top $\tau = 0.35$ fraction of the population, following iAMaLGaM. When continuous variables are sampled, they are always accepted without any other restrictions. The discrete model is built from the entire population. To generate selection pressure, when a mask is applied, the resulting solution is only accepted if it improves, or is equal to the solution following LTGA.

<p>Mixed-Integer Hybrid EA for $i \in \{0, 1, \dots, n - 1\}$ do $\mathcal{P}_i \leftarrow \text{CREATERANDOMSOLUTION}()$ $\text{EVALUATEFITNESS}(\mathcal{P}_i)$</p> <p>while $\neg \text{TERMINATIONCRITERIONSATISFIED}$ do $\text{LEARNDISCRETEMODEL}(\mathcal{P})$ for $i \in \{0, 1, \dots, 2l_d - 1\}$ do $\mathcal{S} \leftarrow \text{TRUNCATIONSELECTION}(\mathcal{P}, \tau)$ $\text{LEARNCONTINUOUSMODEL}(\mathcal{S})$ for $j \in \{0, 1, \dots, n - 1\}$ do $\mathcal{X}_i \leftarrow \text{GENERATECONTINUOUSPART}(\mathcal{P}_i)$ $\mathcal{X}_i \leftarrow \text{GENERATEDISCRETEPART}(j, \mathcal{X}_i, \mathcal{P})$ $\mathcal{P} \leftarrow \mathcal{X}$</p>	<p>GENERATECONTINUOUSPART(\mathcal{P}_i) $\mathcal{X}_c \leftarrow \text{SAMPLECONTINUOUSMODEL}()$ $\mathcal{X} \leftarrow \mathcal{X}_c \cup \mathcal{P}_{i_d}$ $\text{EVALUATEFITNESS}(\mathcal{X})$ return \mathcal{X}</p> <hr/> <p>GENERATEDISCRETEPART($j, \mathcal{X}_i, \mathcal{P}$) $\mathcal{X}_{prev} \leftarrow \mathcal{X}_i$ $donor \leftarrow \text{GETRANDOMSOL}(\mathcal{P})$ $\mathcal{X}_d \leftarrow \text{COPYSUBSET}(j, donor, \mathcal{X}_i)$ $\mathcal{X} \leftarrow \mathcal{X}_{i_c} \cup \mathcal{X}_d$ $\text{EVALUATEFITNESS}(\mathcal{X})$ if $fitness(\mathcal{X}) \geq fitness(\mathcal{X}_{prev})$ then return \mathcal{X} else return \mathcal{X}_{prev}</p>
--	---

Fig. 1. Pseudo-code for generating solutions for mixed integer problems with the integrated version of the LTGA and iAMaLGaM Learning Models

4 Experimental Results

4.1 Benchmark Problems

To design the MI benchmarks we use some well-established benchmark problems and adapt them into the MI setting. In all problems, minimization is assumed. Definitions of the well-established benchmark functions can be found in Table 1. Note that due to minimization, zero is the optimal value for all our functions.

The Sphere function is a very simple continuous function, where all variables are completely independent. Rotated Ellipsoid is a stretched version of the Sphere function. The **R** matrix rotates all variables by 45 degrees, creating dependencies between all continuous variables. In the discrete domain, Onemax is arguably the simplest discrete function, with no parameter dependencies. A Deceptive Trap function is a dependent discrete function. The DT5 function we use here is a non-overlapping, additively decomposable composition of the well-known deceptive trap function, with order $k=5$.

Table 1. Continuous and Discrete functions which are used to define our MI benchmarks

Function Name	Domain	Definition
Sphere	Continuous	$F_{Sphere}(X_c) = \sum_{i=0}^{l_c-1} c_i^2$
Rotated Ellipsoid	Continuous	$F_{R.Ellip.}(X_c) = F_{Ellip.}(\mathbf{R} * X_c)$, where $F_{Ellip.}(X_c) = \sum_{i=0}^{l_c-1} 10^{6*i/(l_c-1)} * c_i^2$
Onemax	Discrete	$F_{Onemax}(X_d) = \sum_{i=0}^{l_d-1} d_i$
Deceptive Trap	Discrete	$F_{DT5}(X_d) = \sum_{i=0}^{l_d/k-1} f_{Trap-k}^{sub}(\sum_{j=ki}^{ki+k-1} d_j)$, where $f_{Trap-k}^{sub} = \begin{cases} 0 & : \text{if } u = k \\ 1 - (k - 1 - u)/k & : \text{otherwise} \end{cases}$

Independently Mixed Benchmarks. We consider all combinations of discrete and continuous problems where the contributions of the discrete and continuous parts are kept independent through addition, see Table 2. Variables in F_1 are fully independent. Only continuous variables are dependent in F_2 . Only discrete variables are dependent in F_3 . In F_4 both sub-spaces are dependent.

Table 2. $F_1 - F_4$: Domain Independent MI Benchmarks

ID	Function name	Definition
F_1	OnemaxSphere	$F_1(X_d, X_c) = F_{Onemax}(X_d) + F_{Sphere}(X_c)$
F_2	Rotated Ellipsoid	$F_2(X_d, X_c) = F_{Onemax}(X_d) + F_{R.Ellip.}(X_c)$
F_3	DT5Sphere	$F_3(X_d, X_c) = F_{DT5}(X_d) + F_{Sphere}(X_c)$
F_4	DT5Ellipsoid	$F_4(X_d, X_c) = F_{DT5}(X_d) + F_{R.Ellip.}(X_c)$

Cross-Domain Dependence Benchmark. The first four of our proposed benchmark problems keep the dependencies within either continuous, discrete or both parameter sub-spaces. The F_5 benchmark includes cross-domain dependencies between the continuous and discrete variables. It is a specific combination of the previously defined F_{DT5} function with the rotated ellipsoid. It is additively decomposable and consists of sub-functions pertaining to blocks of k discrete and k continuous variables.

More specifically, for a trap function with $k = 5$, there are $2^k = 32$ different binary combinations per block. A differently translated rotated ellipsoid function corresponds with each of those combinations (the origin of each function was randomly generated in $[-5,5]$). This way, the continuous function which is being optimized depends on the binary counterpart, introducing dependencies between the discrete and continuous variables that pertain to the same subset. In this benchmark the number of discrete variables is the same as continuous variables: $l_d = l_c = l/2$.

$$F_5(X_d, X_c) = \sum_{i=0}^{0.5l/k-1} (1 + 10^a f_{sub}^{trap}(\sum_{j=ki}^{ki+k-1} d_j)) * (1 + f_{Ellipse}^{sub}(D_i^{block}, C_i^{block})),$$

where D_i^{block} is a block of five discrete variables, and C_i^{block} are the corresponding five real variables. The D block variables determine which of the 2^k different ellipsoid functions need to be optimized, while the C block provides the values of the ellipsoid function

variables. The a value acts as a scaling factor, changing the scale of contribution from the trap function to the overall fitness. In order to solve this benchmark, the algorithm needs to not only solve the trap function, but also optimize the correct rotated ellipsoid function.

4.2 Results on Domain Independent Problems

Heat Maps. We compute the population size that corresponds to the minimal total evaluations needed to solve a problem. The results shown are based on the population sizes for which each problem was solved with the precision of 10^{-10} at least 29/30 times with least evaluations. The results were obtained via bisection.

For the heat maps and scalability analysis of $F_1 - F_4$, we consider different problem lengths: 40, 80, 120 and 160 total variables. For each of these problem sizes, we consider different proportions of variables used with 5, .25*l*, .5*l*, .75*l* and $l - 5$ continuous variables (and $l - l_c$ corresponding discrete variables).

The heat maps in Figure 2 show how the proportions of variable types (discrete or continuous) affects algorithmic efficiency in terms of population sizes and evaluations required to solve the problem.

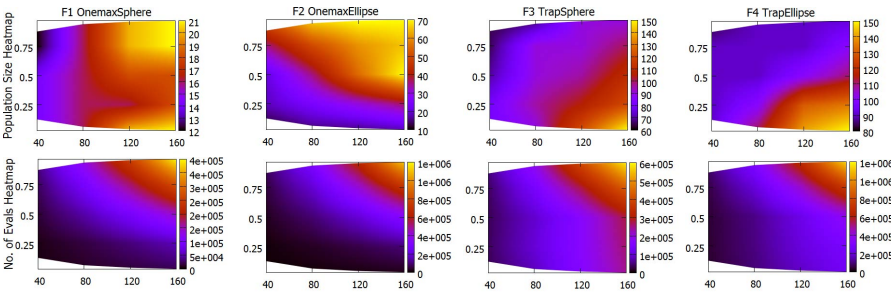


Fig. 2. Heat Maps representing the population sizes (top row) and evaluations (bottom row) needed for different variable compositions. Horizontal axis represents the problem length, the vertical axis is the fraction of continuous variables ($l_c/(l_c + l_d)$) in the problem.

Intuitively F_1 should be the simplest problem for the algorithm to handle, as it contains no parameter dependencies. As the problem composition shifts towards more continuous landscape, the algorithm requires more evaluations. The required population sizing is less affected by the problem composition for F_1 than for the remaining benchmarks. This can be explained by the simplicity and independence of all problem variables.

The effects of changing the problem composition strengthen when partial dependencies are introduced into the problem landscape. As with F_1 , for the remaining benchmarks more evaluations are also required for the same problem sizes as the composition of the problem shifts towards larger numbers of continuous variables. Moreover, as expected, benchmarks which contain dependencies within the continuous sub-space, F_2 and F_4 , require larger number of evaluations than F_1 or F_3 .

Population sizes are also affected by the problem composition. In F_3 and F_4 we observe much larger population size requirements, as the landscape of these functions includes discrete variable dependencies.

This shows that in addition to problem length, the composition of the problem and variable dependencies are a big factor for efficiency in terms of evaluations and population sizes.

Scalability. Figure 3 demonstrates changes in scalability of population size and evaluations over benchmarks $F_1 - F_4$ when the proportions of discrete to continuous variables is changed. Results are shown on a log-log scale. This means that polynomial scalability is indicated by straight lines. From the scalability graphs it is clear that factors such as variable ratios and dependencies can strongly affect the behavior of our algorithm. As expected, the results exhibit polynomial scalability on the tested MI problems.

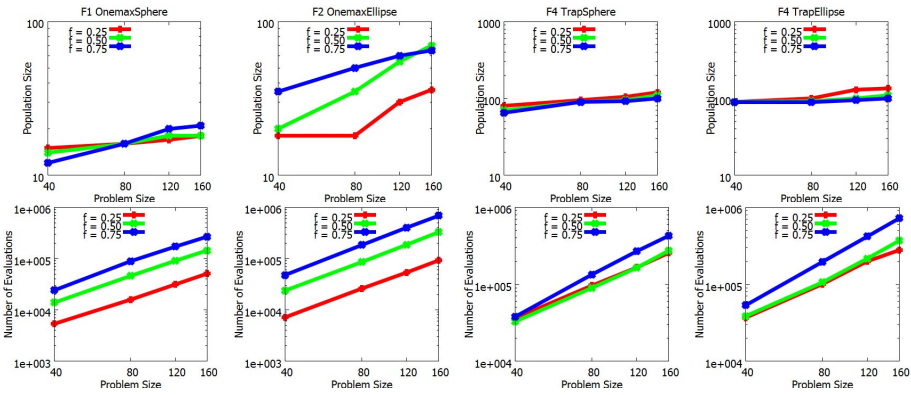


Fig. 3. Scalability of population size and evaluations required for benchmarks $F_1 - F_4$. Fraction f represents the fraction of continuous variables in the problem with length l .

Table 3 shows linear least squares regressions on log-log-scaled data for minimal average number of evaluations e and corresponding population sizes n depending on the problem length l and error term ϵ as follows:

$$\log(n) = \log(l^\alpha) + \epsilon \quad \text{and} \quad \log(e) = \log(l^\beta) + \epsilon.$$

Table 3 shows that population sizes scale sub-linearly. It also shows that performance in terms of evaluations scales more favorably as the problem shifts towards more discrete variable composition.

The use of incremental estimates of iAMaLGaM is crucial in keeping small population sizes. Maintaining evaluation balance and asynchronous model learning, as described in the earlier section, leads to preventing over-exploration or premature convergence much more efficiently than learning the continuous and discrete models synchronously. Graphs (a) and (b) in Figure 4 verify these observations by showing the improved scalability on F_4 over an approach using the original AMaLGaM and with the generations of both models advancing simultaneously.

Table 3. Regression coefficients for scalability

l_c / l_d	$F_1 \alpha$	$F_2 \alpha$	$F_3 \alpha$	$F_4 \alpha$	$F_1 \beta$	$F_2 \beta$	$F_3 \beta$	$F_4 \beta$
5 / 1-5	0.3089	0.1621	0.4458	0.3840	1.1420	1.0663	1.4376	1.4132
0.25 1 / 0.75 1	0.1286	0.5216	0.2815	0.3146	1.6284	1.8432	1.4017	1.4876
0.51 / 0.5 1	0.1952	0.9152	0.3089	0.1370	1.6972	1.8940	1.5187	1.6252
0.75 1 / 0.25 1	0.4218	0.4557	0.3021	0.0727	1.7456	1.9291	1.7572	1.8815
1-5 / 5	0.3307	0.3307	0.2620	0.0539	1.8381	2.0427	1.8566	1.9880

Some overhead still exists, however. By simply combining a continuous problem with a discrete one into one population and treating it as one MI problem, evaluation overhead is introduced into the algorithm. In graph (b) of Fig. 4, the "Separate" label demonstrates what happens if the discrete and continuous sub-domains are solved separately. This separate approach is more efficient because the optimal population sizes can be chosen individually. Additionally, the fitness function is not affected by the other sub-domain. Of course this separate approach is not possible if cross-domain dependencies are present in the problem.

We also consider an alternative approach based on Evolution Strategies (ES). MIES [5] is an ES which extends $(\mu + \lambda)$ -ES for continuous problems to the mixed-integer search spaces. This approach applies a recombination operator, followed by a mutation operator for every solution. Those operators are defined differently for continuous, nominal discrete and integer problem variables. This procedure repeats until λ offspring solutions are created. Best μ solutions from the union of the μ parent solutions and the λ offspring are selected and carried over into the population $P(t + 1)$ [6]. The MIES algorithm has already been shown to work efficiently on some specific industrial problems as well as a set of general MI benchmarks [5]. In graph (c) of Figure 4 we show the performance of MIES on the F_1 benchmark. While it shows good scalability on this simple benchmark, MIES was not able to solve the remaining $F_2 - F_4$ benchmarks within our experimental limits. We conclude that due to existing variable dependencies in $F_2 - F_4$ MIES is not able to solve them efficiently, and requires additional mechanisms in order to handle such MI problems.

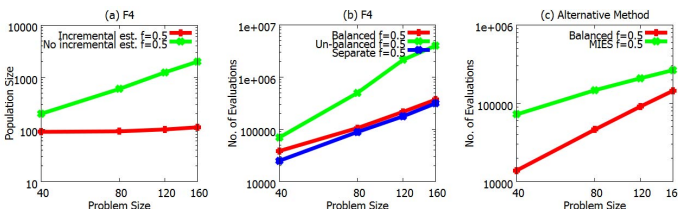


Fig. 4. Importance of (a) incremental estimates and (b) evaluation balancing on F_4 . (c) MIES performance on F_1 .

4.3 Results on the Dependent Problem

Experimental results on F_5 show that the hybrid algorithm we propose, which integrates two independent models is in fact capable of solving this fully dependent benchmark, however not without encountering another important obstacle which is unique to MI problems, and which did not manifest itself as strongly in the other benchmarks. In this problem, blocks of discrete variables control which continuous function needs to be optimized with counterpart continuous variables. The discrete variables, as well as continuous variables, contribute to the overall fitness. This means that in order to find the global optimum, the best discrete variable assignment has to be found, and the continuous function mapped to this assignment must be optimized. The problem arises in the actual numerical values of fitness contributions from either the discrete or continuous sides. The cumulative fitness value of F_5 can be very deceiving depending on the actual differences in scale of fitness contributions. In a regular, strictly discrete Deceptive Trap function the actual fitness values are irrelevant - as long as the function remains deceptive. This is no longer the case in a mixed-integer setting, where the total fitness relies on the contribution from the continuous domain as well. If the trap values are very small in comparison with the continuous variables fitness contributions it becomes more difficult to optimize the discrete variables as they only appear as irrelevant noise to the evaluation function. On the other hand, if the trap values are significantly larger than ones from the continuous subspace, the problem becomes simpler. This behavior is illustrated in Figure 5.

As shown in the definition of F_5 , a controls the scaling of the actual values of the trap function. The larger a , the larger the fitness contribution of the deceptive trap values. In essence, the larger a , the more important it is for the algorithm to solve the trap function of F_5 . Figure 5 demonstrates how much impact this factor has on the success of the algorithm. We were not able to consistently solve F_5 for a values < 1.1 . For these values, the trap function fitness contributions are initially very small, resulting in the algorithm prematurely converging on sub-optimal solutions. As a increases, the problem becomes simpler and requires smaller population sizes and less evaluations.

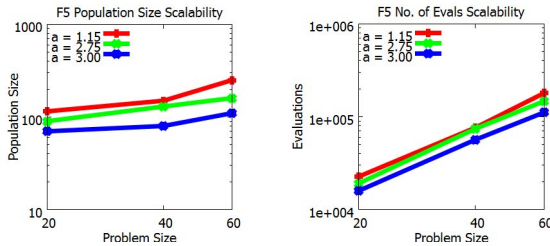


Fig. 5. Scalability on F_5 for different values of a

5 Discussion and Conclusions

Mixed-Integer problems introduce many optimization challenges which do not arise in purely real or discrete optimization problems. With the use of carefully designed

benchmarks, we were able to identify some of such challenges. Obtaining a proper balance in exploration of model information for different types of variables, varying variable ratios and additional overhead or fitness contribution scaling are some of the important issues which should be taken into account when solving MI problems. We made use of two algorithms: LTGA and iAMaLGaM, which are state-of-the-art model-based EAs for problems in discrete and continuous spaces respectively. By extracting key features from these algorithms and carefully integrating the two different models, we were able to study and solve mixed integer benchmarks with varying degrees of variable dependencies. The resulting algorithm achieved polynomial scale-up behavior on the tested benchmarks. We showed that a well-balanced algorithm can solve even very dependent mixed-integer problems, despite having independent model learning methods for the discrete and continuous sub-spaces. The results provide a good foundation and motivation for further work in mixed-integer landscapes with model building EAs. The existing MI EA known as MIES was not able to solve most of our benchmark problems.

While it is very interesting to see that an independent learning approach is capable of solving strongly dependent MI problems, we also learned that this approach has its limitations. Problems with dependencies between the continuous and discrete sub-spaces can be troublesome to an approach using independent learning models. This was shown on the F_5 where for some values of a the algorithm did not succeed. This motivates further work into fully integrating the discrete and continuous models in order to allow learning of cross-domain dependencies, making it possible to solve a greater range of dependent problems.

References

1. Bosman, P.A.N., Grahl, J., Thierens, D.: Enhancing the Performance of Maximum-Likelihood Gaussian EDAs Using Anticipated Mean Shift. In: PPSN, pp. 133–143 (2008)
2. Bosman, P.A.N., Grahl, J., Thierens, D.: AMaLGaM IDEAs in noiseless black-box optimization benchmarking. In: GECCO (Companion), pp. 2247–2254 (2009)
3. Emmerich, M., Grötzner, M., Groß, B., Schütz, M.: Mixed-Integer Evolution Strategy for Chemical Plant Optimization with Simulators. In: Parmee, I.C. (ed.) *Evolutionary Design and Manufacture*, pp. 55–67. Springer, London (2000)
4. Emmerich, M.T.M., Li, R., Zhang, A., Flesch, I., Lucas, P.: Mixed-Integer Bayesian Optimization Utilizing A-Priori Knowledge on Parameter Dependences. In: BNAIC 2008, pp. 65–72 (2008)
5. Li, R., Emmerich, M.T.M., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J., Reiber, J.H.C.: Mixed Integer Evolution Strategies for Parameter Optimization. *Evolutionary Computation* 21(1), 29–64 (2013)
6. Runarsson, T., Yao, X.: Constrained evolutionary optimization. In: *Evolutionary Optimization. International Series in Operations Research and Management Science*, vol. 48, pp. 87–113. Springer, US (2002)
7. Thierens, D.: The linkage tree genetic algorithm. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 264–273. Springer, Heidelberg (2010)
8. Thierens, D., Bosman, P.A.N.: Optimal mixing evolutionary algorithms. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pp. 617–624. ACM, New York (2011)

A New EDA by a Gradient-Driven Density

Ignacio Segovia Domínguez, Arturo Hernández Aguirre, and S. Ivvan Valdez

Center for Research in Mathematics, Guanajuato, México
{ijsegoviad, artha, ivvan}@cimat.mx

Abstract. This paper introduces the Gradient-driven Density Function ($\nabla_d D$) approach, and its application to Estimation of Distribution Algorithms (EDAs). In order to compute the $\nabla_d D$, we also introduce the Expected Gradient Estimate (EGE), which is an estimation of the gradient, based on information from other individuals. Whilst EGE delivers an estimation of the gradient vector at the position of any individual, the $\nabla_d D$ delivers a statistical model (e.g. the normal distribution) that allows the sampling of new individuals around the direction of the estimated gradient. Hence, in the proposed EDA, the gradient information is inherited to the new population. The computation of the EGE vector does not need additional function evaluations. It is worth noting that this paper focuses in black-box optimization. The proposed EDA is tested with a benchmark of 10 problems. The statistical tests show a competitive performance of the proposal.

Keywords: Gradient estimation, Estimation of Distribution Algorithm.

1 Introduction

Several Evolutionary Algorithms search the global optimum by simulations from statistical models; e.g. EDAs, ES, etc. The evolutionary computation community has been making a large effort to add new information into statistical models in order to improve the search process. There are several approaches to add search directions into statistical models [3] [2]. In this context, some popular algorithms have demonstrated the feasibility of this idea (e.g. CMA-ES, NES, etc.). *This paper introduces contributions in this trend by building density functions based on gradient estimates: Gradient-driven densities.* The proposal developed here only use the function evaluations gathered from the population to build gradient estimates on fixed individuals. Hence, the algorithm does not require any extra evaluation of function. The first-order information is an important source of promising directions to improve any individual. For that reason, a Gradient-driven Density Function ($\nabla_d D$) is introduced. Any simulation from $\nabla_d D$ might produce samples around the gradient estimation. Hence, the search process focuses in promising orientations. These novel ideas are merged to create a new EDA. As a consequence of the gradient estimation, the proposed EDA generates new individuals towards promising regions. The organization of the paper is as follows. Section 2 introduces the Expected Gradient Estimation method. Section 3 develops the Gradient-driven Density framework. Section 4 presents the

EDA based on Gradient-driven Density Functions. Section 5 is devoted to test the proposed EDA against others algorithms from the state of the art. Finally, Section 6 provides some concluding remarks.

2 The Gradient Estimation

The gradient vector $\nabla\mathcal{F}(\mathbf{x})$ models the local greatest rate of increase by specifying a direction and magnitude at \mathbf{x} . Since the information about the problem comes from scattered samples on the search space, a neighborhood for an individual might be chosen. For that reason, this paper considers that any gradient estimate for individual $\mathbf{x}^{(i)}$ requires itself and its neighborhood, i.e. a set of individuals $\mathcal{N}_{\mathbf{x}^{(i)}} = \{\mathbf{x}^{(i_1)}, \mathbf{x}^{(i_2)}, \dots, \mathbf{x}^{(i_k)}, \dots, \mathbf{x}^{(i_{r-1})}, \mathbf{x}^{(i_r)}\}$ from the population or gathered from previous generations, where $i \neq i_1 \neq \dots \neq i_k \neq \dots \neq i_r$ and r is the neighborhood size. Furthermore, any criterion to select the neighborhood can be used. Also, notice that this method permits to compute a gradient estimate for each individual only by using known information about the problem. The fitness values of $\{\mathbf{x}^{(i)}, \dots, \mathbf{x}^{(i_1)}, \dots, \mathbf{x}^{(i_r)}\}$ provide knowledge about the problem. Hence, that information can be used to estimate the gradient vector of $\mathbf{x}^{(i)}$. The common approach approximates the gradient by fitting a hyperplane in $d + 1$ dimensions, where d is the dimension size of $\mathbf{x}^{(i)}$. Therefore, the estimation of gradient might be tackled by the ordinary least square method [4]. Despite the fact that the previous technique creates an intuitive gradient approximation, in many contexts it might be inadequate (e.g. there are not enough samples to create the overdetermined system, etc). This section presents a new gradient estimation based on two mathematical concepts: *the directional derivative and the statistical expectation*.

Definition 1. Let $\mathcal{N}_{\mathbf{x}^{(i)}} = \{\mathbf{x}^{(i_1)}, \mathbf{x}^{(i_2)}, \dots, \mathbf{x}^{(i_k)}, \dots, \mathbf{x}^{(i_{r-1})}, \mathbf{x}^{(i_r)}\}$ be the neighbors of individual $\mathbf{x}^{(i)}$, from the population. Then the Expected Gradient Estimate for $\mathbf{x}^{(i)}$ is defined by

$$\widehat{\nabla\mathcal{F}}(\mathbf{x}^{(i)}) = \frac{1}{r} \sum_{k=1}^r \frac{\mathcal{F}(\mathbf{x}^{(i_k)}) - \mathcal{F}(\mathbf{x}^{(i)})}{\|\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}\|^2} (\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}) \tag{1}$$

where $i \neq i_1 \neq \dots \neq i_k \neq \dots \neq i_r$, r is the neighborhood size and $\mathcal{F}(\cdot)$ computes the fitness value.

In order to justify equation (1), let us assume that $\mathbf{x}^{(i_k)}$ exists on the line defined by the true gradient $\nabla\mathcal{F}(\mathbf{x}^{(i)})$. This means

$$\frac{\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}}{\|\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}\|} = \pm \frac{\nabla\mathcal{F}(\mathbf{x}^{(i)})}{\|\nabla\mathcal{F}(\mathbf{x}^{(i)})\|}. \tag{2}$$

Since the orientation depends on the sign, each case will be examined separately. Let \mathbf{u}_+ and \mathbf{u}_- be two normalized vectors as follows

$$\mathbf{u}_+ = \frac{\nabla\mathcal{F}(\mathbf{x}^{(i)})}{h}, \quad \mathbf{u}_- = -\frac{\nabla\mathcal{F}(\mathbf{x}^{(i)})}{h} \tag{3}$$

where $h = \|\nabla\mathcal{F}(\mathbf{x}^{(i)})\|$; please note \mathbf{u}_+ has the same direction as the true gradient, opposite to \mathbf{u}_- . From the well-known directional derivative definition and its properties observe that

$$\left(\lim_{h \rightarrow 0} \frac{\mathcal{F}(\mathbf{x}^{(i)} + h\mathbf{u}_+) - \mathcal{F}(\mathbf{x}^{(i)})}{h}\right) \mathbf{u}_+ = \|\nabla\mathcal{F}(\mathbf{x}^{(i)})\| \frac{\nabla\mathcal{F}(\mathbf{x}^{(i)})}{\|\nabla\mathcal{F}(\mathbf{x}^{(i)})\|} = \nabla\mathcal{F}(\mathbf{x}^{(i)})$$

$$\left(\lim_{h \rightarrow 0} \frac{\mathcal{F}(\mathbf{x}^{(i)} + h\mathbf{u}_-) - \mathcal{F}(\mathbf{x}^{(i)})}{h}\right) \mathbf{u}_- = -\|\nabla\mathcal{F}(\mathbf{x}^{(i)})\| \frac{-\nabla\mathcal{F}(\mathbf{x}^{(i)})}{\|\nabla\mathcal{F}(\mathbf{x}^{(i)})\|} = \nabla\mathcal{F}(\mathbf{x}^{(i)})$$

This is important because similar connections can be found just by considering two individuals, mainly due to the assumption in equation (2); for instance

$$\left(\lim_{l \rightarrow 0} \frac{\mathcal{F}(\mathbf{x}^{(i)} + l \cdot \mathbf{u}) - \mathcal{F}(\mathbf{x}^{(i)})}{l}\right) \mathbf{u} = \nabla\mathcal{F}(\mathbf{x}^{(i)}) \tag{4}$$

$$\mathbf{u} = \frac{\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}}{\|\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}\|}, \quad l = \|\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}\| \tag{5}$$

Equation (4) presents a different manner to remake the gradient function. Notice that although the derivative is unavailable, an approximation by finite differences can be considered. It leads us to introduce a gradient estimate for $\mathbf{x}^{(i)}$, just given *one* neighbor, as follows

$$\mathbf{g}^{(i_k)} = \left(\frac{\mathcal{F}(\mathbf{x}^{(i)} + l\mathbf{u}) - \mathcal{F}(\mathbf{x}^{(i)})}{l}\right) \mathbf{u} = \frac{\mathcal{F}(\mathbf{x}^{(i_k)}) - \mathcal{F}(\mathbf{x}^{(i)})}{\|\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}\|^2} (\mathbf{x}^{(i_k)} - \mathbf{x}^{(i)}) \tag{6}$$

However, there is no chance to ensure that $\mathbf{x}^{(i_k)}$ is on the line defined by the true gradient, because the neighbors come from an unknown hidden random process. Hence, the difference between fitness values is also a random variable. Therefore, each estimate $\mathbf{g}^{(i_k)}$ arises from a random process. Please assume that \mathcal{P} is the hidden uncertainty model which describes the behavior of outcomes $\mathbf{g}^{(i_k)}$. So, any instance of random variable $\mathbf{g}^{(i)} \sim \mathcal{P}$ is an outcome $\mathbf{g}^{(i_k)}$. A representative vector for the hidden model can be computed by the statistical expectation. Moreover, the $E(\mathbf{g}^{(i)})$ can be approximated as follows

$$E(\mathbf{g}^{(i)}) = \int_{\mathbb{R}^d} \mathbf{g}^{(i)} \mathcal{P} d\mathbf{g}^{(i)} \approx \frac{1}{r} \sum_{k=1}^r \mathbf{g}^{(i_k)} \tag{7}$$

which is the *Expected Gradient Estimate*, equation (1).

To the best of our knowledge, the EGE developed above has not been addressed in literature. However, further theoretical study is necessary to verify its relationship with other approaches [1]. In order to empirically contrast the approximated orientations of EGE versus the usual approximation by hyperplane, a fixed population and a gradient estimation on each individual will be considered. An ideal population, at first generation, must cover the search domain

evenly; thus in this experiment the population is built by the Halton quasi-random sequence, from Matlab® with default options. Also, please consider the Sphere problem, the neighborhood size $r = d + 1$ and the r closer individuals to $\mathbf{x}^{(i)}$ (neighborhood, according to the Euclidean distance). Below there is an angle-comparison between $\widehat{\nabla\mathcal{F}}(\mathbf{x}^{(i)})$ and $\nabla\mathcal{F}(\mathbf{x}^{(i)})$. So, the measurement vector of angles $\boldsymbol{\alpha} = \{\alpha^{(1)}, \dots, \alpha^{(i)}, \dots, \alpha^{(N)}\}$ includes an angle value for each individual, where N is the population size. Figure 1 shows the histograms of orientation by

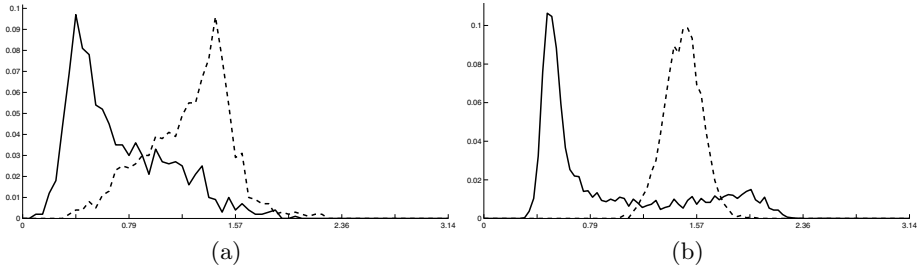


Fig. 1. Histograms of orientation in Sphere problem, Expected Gradient Estimation (EGE, solid line) versus hyperplane approach by ordinary least square (HLS, dashed line). (a) 10 dimensions: HLS has a median value of 1.30488 and EGE has a median value of 0.627966. (b) 30 dimensions: HLS has a median value of 1.48743 and EGE has a median value of 0.617965. EGE shows better performance.

setting $r = d + 1$, $N = 10d$ and $x_k \in [-600, 300]$; where d is the dimension size. Notice that the EGE has more chances to compute better oriented vectors than the hyperplane approach for higher dimensions. In addition, the median values support the graphic observation. Here, the neighborhood size $r = d + 1$ was chosen from literature [4]. In summary, the results suggest that the EGE might outperform previous gradient approximations used in evolutionary computation.

3 The Gradient-Driven Density

Each generation has three different data sets: the individuals $\{\mathbf{x}^{(i)}\}$, the function evaluations $\{\mathcal{F}(\mathbf{x}^{(i)})\}$ and the estimations of the gradient estimates $\{\widehat{\nabla\mathcal{F}}(\mathbf{x}^{(i)})\}$. These provide distinct information about the function and the algorithms' behavior. Several stochastic optimization approaches (e.g. ES, EDA) aim to build a multivariate density function on optimum locations; or at least in better regions than the current ones. This section will begin with the same goal for a fixed individual from the population. Therefore, there might exist a multivariate density function $p(\mathbf{x}, \boldsymbol{\theta})$ for $\mathbf{x}^{(i)}$ based on its gradient estimate $\widehat{\nabla\mathcal{F}}(\mathbf{x}^{(i)})$, which is able to simulate better individuals than the present $\mathbf{x}^{(i)}$. Due to the fact that only two vectors will be used here, there is no chance to ensure that all simulations improve the current fitness value $\mathcal{F}(\mathbf{x}^{(i)})$. However, the density modeling

can be modified to take advantage of the gradient estimate by developing a new estimation of parameters, updating the original parameters, improving the simulation, etc. For this reason, definition 2 introduces the $\nabla_d D$ from the individual perspective.

Definition 2. Let \mathbf{z} be an individual in the domain space and $G(\mathbf{z})$ a function which computes its gradient estimate. The density function $p(\mathbf{x}, \boldsymbol{\theta})$ is a Gradient-driven Density ($\nabla_d D$) for individual \mathbf{z} if the following two conditions are satisfied:

- 1) The multivariate density $p(\mathbf{x}, \boldsymbol{\theta})$ is a unimodal function,
- 2) $\frac{G(\mathbf{z})}{\|G(\mathbf{z})\|} = \frac{\nabla p(\mathbf{z}, \boldsymbol{\theta})}{\|\nabla p(\mathbf{z}, \boldsymbol{\theta})\|}$.

The conditions set up above allow a wide range of future proposals. The first condition permits a single mass of probability towards promising regions. The random search must be led by $G(\mathbf{z})$ because it is orienting towards promising regions. In fact, the second condition just allows density functions which $\nabla p(\mathbf{z}, \boldsymbol{\theta})$ has the same direction as the gradient estimate $G(\mathbf{z})$. There are many ways to build a ∇_d Density. Below, a suitable technique based on multivariate calculus and the angular discrepancy are introduced .

Definition 3. Let $p(\mathbf{x}, \boldsymbol{\theta})$ be a multivariate unimodal density. In order to ensure that $p(\mathbf{x}, \boldsymbol{\theta})$ is a ∇_d Density, a parameter estimation on $\boldsymbol{\theta}$ must be performed. The minimum angle estimation solves this by,

$$\hat{\boldsymbol{\theta}} = \max_{\boldsymbol{\theta}} \frac{G(\mathbf{z})^t \nabla p(\mathbf{z}, \boldsymbol{\theta})}{\|G(\mathbf{z})\| \|\nabla p(\mathbf{z}, \boldsymbol{\theta})\|} \tag{8}$$

Notice that the *minimum angle estimation* solves the parameter estimation of $\boldsymbol{\theta}$ by maximizing the dot product of two normalized vectors. It is a natural way because the angle between two vectors is related to the dot product. In addition, even if finding the solution of (8) is not possible, a good approximation can be discovered. The rest of this section uses the previous definition to build ∇_d densities. Please assume that $p(\mathbf{x}, \boldsymbol{\theta})$ is a *multivariate normal density* and $\nabla p(\mathbf{z}, \boldsymbol{\theta})$ its gradient function. Also, let

$$\mathbf{z}_G = \frac{G(\mathbf{z})}{\|G(\mathbf{z})\|} \tag{9}$$

be the normalized gradient estimate of individual \mathbf{z} . The estimation method must calculate the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ which satisfy definition 2. Then, the statistical parameters can be found by solving

$$\langle \boldsymbol{\mu}^{new}, \boldsymbol{\Sigma}^{new} \rangle = \max_{\langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle} \frac{\mathbf{z}_G^t [\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \mathbf{z})]}{\|\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \mathbf{z})\|}. \tag{10}$$

Let us address this problem separately for each parameter. By taking the derivative with respect to $\boldsymbol{\mu}$ and setting it equal to zero, we found the equation

$$\|\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z})\|^2 \boldsymbol{\Sigma}^{-t} \mathbf{z}_G - \mathbf{z}_G^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z}) \boldsymbol{\Sigma}^{-t} \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z}) = 0. \quad (11)$$

In a similar way, by taking the derivative with respect to $\boldsymbol{\Sigma}^{-1}$ and setting it equal to zero, we found the equation

$$\|\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z})\|^2 \mathbf{z}_G (\boldsymbol{\mu} - \mathbf{z})^t - \mathbf{z}_G^t \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z}) \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z}) (\boldsymbol{\mu} - \mathbf{z})^t = 0. \quad (12)$$

Notice that both are nonlinear matrix equations! In addition, the problem (12) is a constraint equation, since $\boldsymbol{\Sigma}$ needs to be a symmetric positive semidefinite matrix. So, it leads us to solve two more complex optimization problems than the original one. However, a few interesting facts arise by inspecting equations (10)-(12), when $\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z}) = \mathbf{z}_G$: Equation (10) reaches its maximum value, i.e. 1; and Equations (11) and (12) are fulfilled. Furthermore, notice that $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are closely related. In fact, there are an infinite number of symmetric semipositive definite matrices $\boldsymbol{\Sigma}$ able to fulfill $\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z}) = \mathbf{z}_G$. This certainly means that the nonlinear system has an infinity number of solutions. However, straightforward solutions can be found by these observations. By assuming the matrix $\boldsymbol{\Sigma}$ is fixed and solving for the mean vector in $\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{z}) = \mathbf{z}_G$ a new formula is found:

$$\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}^{new} - \mathbf{z}) = \mathbf{z}_G \quad \therefore \quad \boldsymbol{\mu}^{new} = \mathbf{z} + \boldsymbol{\Sigma} \mathbf{z}_G. \quad (13)$$

Given a fixed covariance matrix, its related mean vector can be computed by (13). Furthermore, there is a unique mean vector for a given $\boldsymbol{\Sigma}$. On the contrary, given a fixed mean vector, there is a number of infinite possible covariance matrices. Definition 4 introduces a ∇_d Density based on the previous analysis.

Definition 4. *Let $\boldsymbol{\Sigma}_0$ be a fixed covariance matrix. Then the ∇_d Normal ($\nabla_d \mathcal{N}$) has parameters*

$$\boldsymbol{\mu}_g = \mathbf{z} + \boldsymbol{\Sigma}_0 \mathbf{z}_G \quad \text{and} \quad \boldsymbol{\Sigma}_g = \boldsymbol{\Sigma}_0 \quad (14)$$

Notice that, simulations from the proposed densities will produce samples in a similar direction as the gradient vector (or gradient estimate). The next section applies the developed *Gradient-driven densities* in evolutionary computation.

4 The Gradient-Driven Density in EDAs

The Estimation of Distribution Algorithm (EDA) aims to simulate new individuals on regions near optimum locations, preferably close to the global optimum. Interesting optimization methods might be developed by considering the $\nabla_d \mathcal{D}$ technique into EDAs. The EDA fits a target statistical model. A common one is the multivariate normal function [5]. This section introduces an EDA, based on

this density function, by computing the expectation and variance of a multivariate Gaussian mixture model. Please consider a mixture of two models: the *empirical normal density* and a *Gradient-driven Density*. The first one promotes the exploitation whilst the second one allows predictive samples on possible promising regions (exploration). In order to build a simpler model, the mixture of densities is approximated by a unique Multivariate Gaussian model [6]. The target density for the proposed EDA is built by $\mathcal{N}(\boldsymbol{\mu}^{new}, \boldsymbol{\Sigma}^{new})$, where:

$$\begin{aligned} \boldsymbol{\mu}^{new} &= E(E(\mathbf{X}|\vartheta)) = (1 - \beta)\hat{\boldsymbol{\mu}} + \beta\boldsymbol{\mu}_g, \\ \boldsymbol{\Sigma}^{new} &= Var(E(\mathbf{X}|\vartheta)) + E(Var(\mathbf{X}|\vartheta)) = (1 - \beta)\hat{\boldsymbol{\Sigma}} + \beta\boldsymbol{\Sigma}_g \\ &\quad + (1 - \beta)(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{new})(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}^{new})^t + \beta(\boldsymbol{\mu}_g - \boldsymbol{\mu}^{new})(\boldsymbol{\mu}_g - \boldsymbol{\mu}^{new})^t \end{aligned} \quad (15)$$

and $\beta \in [0, 1]$ is the associated weight to the $\nabla_d D$. Also, β controls the amount of credibility on each model. Please note that $\beta = 0$ produces the empirical density and $\beta = 1$ yields the other one. In addition, since the simulation method might build samples outside the search domain, a re-insertion technique is added, line 10 of algorithm 2. Let $\gamma_k = l_k^{upper} - l_k^{lower}$ be the domain length in dimension k , where l_k^{upper} and l_k^{lower} are the upper bound and lower bound in dimension k . For each dimension, the new sample $\mathbf{y}^{(i)} = (y_1^{(i)}, \dots, y_k^{(i)}, \dots, y_D^{(i)})$ is tested/replaced by

- if $y_k^{(i)} > l_k^{upper}$ then $a = (y_k^{(i)} - l_k^{upper})/\gamma_k$ and $y_k^{(i)} = l_k^{upper} - \gamma_k(a - \lfloor a \rfloor)$
- if $y_k^{(i)} < l_k^{lower}$ then $a = (l_k^{lower} - y_k^{(i)})/\gamma_k$ and $y_k^{(i)} = l_k^{lower} + \gamma_k(a - \lfloor a \rfloor)$

which ensure any new individual is inside the domain. The algorithm 2 describes the proposed EDA led by a Gradient-driven Density (EDA-LGD). Because of the importance of the gradient estimate for the $\nabla_d D$, this proposal just computes the gradient of the best individual using the historical best individuals from previous generations. So, if at generation (t) a new best individual \mathbf{x}^{best} is found, then \mathbf{x}^{best} replaces the worst individual in P_{best} and the next gradient estimate is over \mathbf{x}^{best} with the neighborhood $\{P_{best} \setminus \mathbf{x}^{best}\}$. Then two populations are saved: the usual population Pob_t at each generation (t) and the historically best individuals P_{best} ; in algorithm 2 the first one has N individuals meanwhile the second one has $d + 2$ individuals.

5 Experiment

This section contrasts the proposed EDA against two known Evolutionary Algorithms based on multivariate densities: CMA-ES [3] and xNES [2]. Each algorithm runs in 10 benchmark problems, see Table 1. In order to make a fair comparison, the code was downloaded from authors homepage and 50 runs were performed. Also, the initial center of densities was chosen randomly in the search domain with an initial variance according to the domain (1/3 of this). The three algorithms only have two stopping conditions: maximum number of evaluations of function is reached ($10^4 \times d$), or target error smaller than 10^{-8} ,

```

1:  $t \leftarrow 0, \beta \leftarrow 0.5, N \leftarrow \lceil 4 * (1 + d^{0.7}) \rceil, M \leftarrow 2 * \lfloor \log(d) \rfloor + 1, r \leftarrow d + 1$ 
2:  $Pob_t \leftarrow \mathcal{U}(\text{Domain})$ , compute  $\mathcal{F}(\mathbf{x}^{(i)})$ , find the  $\mathbf{x}^{best}$  ▷ First population
3:  $P_{best} \leftarrow$  Best  $r + 2$  individuals from  $Pob_t$  ▷ Historical best population
4: while (Stop condition is not reached) do
5:   ◦ Gradient estimate  $G(\mathbf{x}^{best}) = \widehat{\nabla} \mathcal{F}(\mathbf{x}^{best})$  with neighborhood  $\{P_{best} \setminus \mathbf{x}^{best}\}$ 
6:   ◦ Normalized vector  $\mathbf{x}_{G^{est}}^{best}$  by (9) or negative for minimization
7:   ◦ Empirical estimation of  $\widehat{\boldsymbol{\mu}}$  and  $\widehat{\boldsymbol{\Sigma}}$ . Initial covariance  $\boldsymbol{\Sigma}_0 = \text{diag}(\text{diag}(\widehat{\boldsymbol{\Sigma}}))$ 
8:   ◦ Parameters  $\boldsymbol{\mu}_g$  and  $\boldsymbol{\Sigma}_g$  by definition 4. Parameters  $\boldsymbol{\mu}^{new}$  and  $\boldsymbol{\Sigma}^{new}$  by (15)
9:   ◦  $\mathcal{S} \leftarrow$  Simulate  $M$  individuals from  $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{new}, \boldsymbol{\Sigma}^{new})$ 
10:  ◦  $\mathcal{S} \leftarrow$  Reinsertion( $\mathcal{S}$ ) ▷ if-outside-domain
11:  ◦ Fitness values  $\mathcal{F}(\mathcal{S})$ 
12:  ◦  $Pob_{t+1} \leftarrow$  Best individuals among  $\{Pob_t, \mathcal{S}\}$ 
13:  ◦ Find the  $\mathbf{x}_{t+1}^{best}$  of  $Pob_{t+1}$ 
14:  if  $\mathbf{x}_{t+1}^{best}$  has better fitness value than  $\mathbf{x}^{best}$  then
15:     $\mathbf{x}^{best} \leftarrow \mathbf{x}_{t+1}^{best}$  and  $\mathbf{x}_{t+1}^{best}$  replaces the worst individual in  $P_{best}$ 
16:  end if
17:  ◦  $M_{sur} \leftarrow$  Number of survivors from  $\mathcal{S}$  into  $Pob_{t+1}$ 
18:  if  $\frac{M_{sur}}{M} > 1/2$  then
19:     $\beta \leftarrow \beta + 0.05$ ; if  $\beta > 1$  then  $\beta = 1$  ▷ Exploration
20:  else
21:     $\beta \leftarrow \beta - 0.05$ ; if  $\beta < 0$  then  $\beta = 0$  ▷ Exploitation
22:  end if
23:   $t \leftarrow t + 1$ 
24: end while

```

Fig. 2. Pseudocode of the EDA led by a Gradient-driven Density (EDA-LGD)

Table 1. Benchmark problems [2] [5]. The minimum fitness value of all problems is 0, except for $\mathcal{F}_4, \mathcal{F}_6$ and \mathcal{F}_{10} where $\mathcal{F}_4^* = 2, \mathcal{F}_6^* = -10$ and $\mathcal{F}_{10}^* = -0.1d$.

Name	Alias	Domain	Name	Alias	Domain
Sphere	\mathcal{F}_1	$x_i \in [-600, 300]$	Different Powers	\mathcal{F}_2	$x_i \in [-20, 10]$
Brown	\mathcal{F}_3	$x_i \in [-1, 4]$	Mishra 2	\mathcal{F}_4	$x_i \in [0, 1]$
Ellipsoid	\mathcal{F}_5	$x_i \in [-20, 10]$	Parabolic Ridge	\mathcal{F}_6	$x_i \in [-20, 10]$
Rosenbrock	\mathcal{F}_7	$x_i \in [-20, 10]$	Ackley	\mathcal{F}_8	$x_i \in [-20, 10]$
Griewangk	\mathcal{F}_9	$x_i \in [-600, 300]$	Negative Cosine Mixture	\mathcal{F}_{10}	$x_i \in [-1, 0.5]$

i.e. $(\mathcal{F} - \mathcal{F}^*) < 10^{-8}$. Figure 3 contrasts the error $\mathcal{F} - \mathcal{F}^*$ reached for each algorithm. Also, this Figure shows a comparison between two algorithms in the second and third columns. For each problem there are three measures: 1) the first row is the percentage of success rate, 2) the second row is the mean and standard deviation of reached fitness values, 3) the third row is the mean and standard deviation of needed evaluations of function. The mean values highlighted with boldface, i.e. the winner algorithm, are supported by a statistical test. The last column presents the results of two nonparametric bootstrap tests. Here, the hypotheses are based on the mean value μ . The hypotheses $(H_0 : \mu_1 \geq \mu_2, H_1 : \mu_1 < \mu_2)$ yields the p-value ρ_1 and $(H_0 : \mu_2 \geq \mu_1, H_1 : \mu_2 < \mu_1)$ produces

\mathcal{F}	EDA-LGD	CMA-ES	ρ_1 vs ρ_2	EDA-LGD	xNES	ρ_1 vs ρ_2
\mathcal{F}_1	100.00	100.00		100.00	100.00	
	8.7e-9±1.2e-9 4.9e+4±1.7e+4	5.5e-9±1.2e-9 3.8e+3±1.4e+2	1.0,1e-4 1.0,1e-4	8.7e-9±1.2e-9 4.9e+4±1.7e+4	8.9e-9±8.7e-10 2.8e+4±2.9e+2	0.2,0.7 1.0,1e-4
\mathcal{F}_2	100.00	100.00		100.00	100.00	
	8.0e-9±1.6e-9 5.5e+3±5.6e+2	9.4e-9±5.9e-10 9.0e+3±7.2e+2	1e-4,1.0 1e-4,1.0	8.0e-9±1.6e-9 5.5e+3±5.6e+2	7.4e-9±2.0e-9 1.6e+4±8.4e+2	0.9,6e-2 1e-4,1.0
\mathcal{F}_3	100.00	100.00		100.00	100.00	
	8.6e-9±1.1e-9 5.1e+3±2.1e+2	5.1e-9±1.3e-9 3.0e+3±1.4e+2	1.0,1e-4 1.0,1e-4	8.6e-9±1.1e-9 5.1e+3±2.1e+2	8.6e-9±1.1e-9 2.4e+4±3.3e+2	0.5,0.4 1e-4,1.0
\mathcal{F}_4	100.00	4.00		100.00	94.00	
	9.4e-9±5.0e-10 2.9e+3±1.2e+2	8.9e-2±8.9e-2 1.9e+5±1.6e+4	1e-4,1.0 1e-4,1.0	9.4e-9±5.0e-10 2.9e+3±1.2e+2	2.4e+2±1.7e+3 8.3e+4±3.1e+4	7e-2,0.9 1e-4,1.0
\mathcal{F}_5	100.00	100.00		100.00	38.00	
	8.6e-9±1.2e-9 9.8e+3±7.9e+2	5.4e-9±1.2e-9 1.8e+4±3.0e+2	1.0,1e-4 1e-4,1.0	8.6e-9±1.2e-9 9.8e+3±7.9e+2	1.7e-3±7.9e-3 1.5e+5±5.8e+4	5e-2,0.9 1e-4,1.0
\mathcal{F}_6	100.00	100.00		100.00	96.00	
	9.0e-9±9.0e-10 7.6e+3±3.9e+2	7.8e-9±1.0e-9 9.3e+3±1.5e+3	1.0,1e-4 1e-4,1.0	9.0e-9±9.0e-10 7.6e+3±3.9e+2	9.3e-9±6.7e-10 5.2e+4±3.3e+4	2e-2,0.9 1e-4,1.0
\mathcal{F}_7	28.00	88.00		28.00	100.00	
	5.5e-1±1.4e+0 1.9e+5±8.7e+3	4.7e-1±1.3e+0 4.2e+4±5.8e+4	0.6,0.3 1.0,1e-4	5.5e-1±1.4e+0 1.9e+5±8.7e+3	8.6e-9±1.2e-9 4.4e+4±2.3e+3	0.9,2e-3 1.0,1e-4
\mathcal{F}_8	82.00	56.00		82.00	100.00	
	2.4e-1±5.6e-1 4.4e+4±7.3e+4	1.5e+0±2.2e+0 9.1e+4±9.7e+4	3e-4,1.0 4e-3,0.9	2.4e-1±5.6e-1 4.4e+4±7.3e+4	9.3e-9±5.5e-10 4.3e+4±4.3e+2	0.9,1e-3 0.5,0.4
\mathcal{F}_9	2.00	76.00		2.00	96.00	
	1.6e+0±2.5e+0 1.9e+5±2.3e+4	2.5e-3±4.9e-3 5.1e+4±8.4e+4	1.0,1e-4 1.0,1e-4	1.6e+0±2.5e+0 1.9e+5±2.3e+4	3.9e-4±2.0e-3 3.2e+4±3.4e+4	1.0,2e-4 1.0,1e-4
\mathcal{F}_{10}	12.00	6.00		12.00	68.00	
	2.7e-1±1.9e-1 1.7e+5±6.4e+4	3.0e-1±1.5e-1 1.8e+5±4.7e+4	0.1,0.8 0.2,0.8	2.7e-1±1.9e-1 1.7e+5±6.4e+4	1.5e-1±5.0e-1 8.7e+4±7.8e+4	0.9,6e-2 1.0,1e-4

(a)

(b)

Fig. 3. Percentage of success rate, reached fitness values and needed number of evaluations (mean and standard deviation) for each algorithm in dimension 20. The last column shows two nonparametric bootstrap tests. If ρ_1 is boldface the winner is EDA-LGD, if ρ_2 is boldface the winner is either CMA-ES or xNES, otherwise there is no winner.

the p-value ρ_2 . So, if ρ_1 is boldface the winner is EDA-LGD, if ρ_2 is boldface the winner is either CMA-ES or xNES, otherwise there is no winner. The null hypothesis is rejected with significance level $\alpha = 0.05$ **Comments (CMAES):** The problems $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_5$ and \mathcal{F}_6 do not seem difficult for EDA-LGD nor CMA-ES, since both algorithms reach the perfect success rate. On the contrary, the rest of the problems have a more difficult landscape. According to the bootstrap test, there is statistical evidence to conclude that in 5 out of 10 problems the proposed EDA requires fewer function evaluations than the CMA-ES. **Comments (xNES):** According to the bootstrap test, there is statistical evidence to conclude that in 5 out of 10 problems the proposed EDA requires fewer function evaluations than the xNES. Also, there appears to be a pattern related to the landscape. For instance, note xNES has better results for problems $\mathcal{F}_7 - \mathcal{F}_{10}$, but EDA-LGD has better results for problems $\mathcal{F}_2 - \mathcal{F}_6$. This kind of pattern must be further studied in future work.

6 Conclusion

This paper presents a new EDA based on the Gradient-driven densities ($\nabla_d D$). In order to build the proposed EDA (EDA-LGD) two main contributions were developed: the Expected Gradient Estimate (EGE) and the $\nabla_d D$. Also, a technique has been proposed to compute a gradient estimate for any individual only by using the actual knowledge about the problem. Hence, the estimation of the gradient does not need extra evaluations of function. The $\nabla_d D$ are statistical models built by taking into account a gradient estimate. This new framework can create a density function for any individual. Consequently, any simulation from those densities has a random gradient component. Here, Gradient-driven densities based on the Multivariate Normal have been constructed. However, the developed framework allows for the assumption of other statistical models. The ideas discussed above motivated a new EDA: EDA-LGD. It is based on the Gradient-driven Independent Normal, the EGE and the hierarchical latent variable model. Moreover, it was tested in 10 benchmark problems; where the EDA-LGD shows competitive performance against CMA-ES and xNES. In summary, the EDA-LGD is an interesting approach because of the performance of the algorithm and its mathematical foundation. Since the $\nabla_d D$ will produce samples in a similar direction as the gradient estimation, this density can be regarded as a predictive model. Thus, the Gradient-Driven density allows for exploration of the search domain whilst the empirical density intends fast convergence (exploitation). Finally, notice that the main contributions developed here can be extended to other evolutionary algorithms.

References

1. Flaxman, A.D., Kalai, A.T., McMahan, H.B.: Online convex optimization in the bandit setting: Gradient descent without a gradient. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, pp. 385–394. Society for Industrial and Applied Mathematics, Philadelphia (2005)
2. Glasmachers, T., Schaul, T., Sun, Y., Wierstra, D., Schmidhuber, J.: Exponential natural evolution strategies. In: Genetic and Evolutionary Computation Conference (2010)
3. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., Larranaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a new evolutionary computation. STUDFUZZ, vol. 192, pp. 75–102. Springer, Heidelberg (2006)
4. Hazen, M., Gupta, M.R.: Gradient estimation in global optimization algorithms. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 1841–1848 (2009)
5. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Boston (2002)
6. Schnatter, F.S.: Finite mixture and Markov switching models. Springer, Heidelberg (2006)

From Expected Improvement to Investment Portfolio Improvement: Spreading the Risk in Kriging-Based Optimization

Rasmus K. Ursem

Research and Technology, Grundfos Management A/S,
Poul Due Jensens Vej 7, 8850 Bjerringbro, Denmark
ursem@cs.au.dk

Abstract. The increasing use of time-consuming simulations in the industry has spawned a growing interest in coupling optimization algorithms with fast-to-compute surrogate models. A major challenge in this approach is to select the approximated solutions to evaluate on the real problem. To address this, the Kriging meta-model offers both an estimate of the mean value and the standard error in an unknown point. This feature has been exploited in a number of so-called prescreening utility functions that seek to maximize the outcome of an expensive evaluation. The most widely used are the Probability of Improvement (PoI) and Expected Improvement (ExI) functions.

This paper studies this challenge from an investment portfolio point-of-view. In short, the PoI favors low risk investments whereas the ExI promotes high risk investments. The paper introduces the investment portfolio improvement (IPI) approach as a strategy mixing the two extremes. The novel approach is applied to seven benchmark problems and two real world examples from the pump industry.

Keywords: Expected improvement, prescreening methods, Kriging.

1 Introduction

During the last couple of decades, the increasing use of time-consuming simulations in engineering-related industries poses a serious challenge to optimization algorithms. To address this challenge, researchers have studied a number of surrogate models that allow fast evaluation of an approximation of the real problem. This approximation is typically build from a low number of sample points of the real problem. Approximation models can be evaluated in a few hundred milliseconds, which is significantly faster than, e.g., a 5 hour flow simulation of a centrifugal pump. However, the result is only an approximation and there are usually differences between the real problem and the approximation of it. Thus, the optimization specialist trades evaluation accuracy for evaluation speed.

The Kriging meta-model has become increasingly popular as it provides both a mean value and standard error of the approximation of an unknown point. By viewing the standard error as a confidence interval for the approximation,

authors have proposed to maximize a *prescreening* function expressing the potential improvement gained by performing the expensive/time-consuming evaluation of the real function. Three widely used prescreening functions are the Lower Bound (LB) approach by Dennis and Torczon [1], the Probability of Improvement (PoI) introduced by Ulmer et al. [2], and the Expected Improvement (ExI) popularized as the Efficient Global Optimization (EGO) algorithm by Jones et al. [3]. The ExI is the most popular and it has been extensively used for solving numerous real-world problems, e.g., [4,5,6]. The focus in this paper is on single-objective problems, but the ExI function has also been adapted to multi-objective problems, e.g., [5,7].

The aim of this paper is twofold. First, to study and discuss LB, PoI, and ExI in relation to their theoretical ability to provide a return-of-investment, i.e., an improved solution to the problem. Second, to suggest an algorithm for optimizing a portfolio of solutions that spread the risk of investment by using the novel prescreening approach called *Investment Portfolio Improvement*. The algorithm is based on the Differential Evolution (DE) algorithm [10,11].

The paper is structured as follows. Section 2 introduces the optimization methodology, in particular the Differential Evolution algorithm and the Kriging meta-model. Subsection 2.3 elaborates on LB, PoI, and ExI in relation to the investment strategy these prescreening functions implement. Subsection 2.4 describes the novel algorithm. Following this, section 3 introduces the experimental setup and the optimization problems. Section 4 contains the results and a discussion of these. Finally, section 5 concludes the paper.

2 Kriging-Based Prescreening Optimization

Successful application of a Kriging-based prescreening optimization algorithm to a time-consuming or costly real-world problem involves three main decisions. First, the choice of the optimization algorithm. Second, the choice of the Kriging variant. Third, the choice of the prescreening function.

Regarding optimization algorithms, a good choice is the differential evolution (DE) algorithm suggested by Storn and Price in 1995 [10,11]. Since then, it has become widely accepted as one of the best algorithms for numerical optimization as it has proven its worth on numerous problems, e.g., [4,8,9].

Concerning Kriging, numerous variants have been described and tested in the literature and choosing the best variant can be a challenge on its own [12,13]. From an optimization perspective, the main choice is a trade-off between accuracy and ability to find a new solution that is better than the best known point. The use of prescreening methods may reduce the disadvantages of choosing a sub-optimal Kriging variant substantially as the methods allow the algorithm to employ an explorative search behavior. Thus, the prescreening method allows a choice of Kriging variant that does not return large overshoots of the best known point. For this reason, this paper uses simple Kriging with a “conservative” kernel function.

2.1 Differential Evolution

The algorithm presented in this paper is based on the *rand/1/bin* standard DE scheme. However, it is out of the scope to provide a detailed description of DE. Instead, see the original work of Storn and Price [10,11] or refer to Ursem [9] for a shorter version.

2.2 Kriging

As mentioned, a large number of Kriging variants exists. Kriging has been described many times in the literature and a full mathematical description is beyond the scope of this paper. Instead, see e.g. [14]. In short, Kriging predicts a normal distribution $Y(\mathbf{x}) \sim N(\hat{y}, \hat{s})$ as an interpolation based on a so-called kernel function of the distances to a number of known points.

The used Kriging model is based on the DACE Matlab toolbox by Lophaven et al. [15] with the EXP kernel function. This kernel function was tested in preliminary runs and showed the desired absence of extreme prediction values.

2.3 Prescreening Procedures

The main idea behind prescreening functions is to utilize the standard error of Kriging to assess the potential improvement achieved by evaluating an unknown point \mathbf{x} . The most widely used prescreening functions are the Lower Bound (LB), the Probability of Improvement (PoI), and the Expected Improvement (ExI).

$$\text{LB}(\mathbf{x}) = \hat{y}(\mathbf{x}) - w \cdot \hat{s}(\mathbf{x}) \quad (1)$$

$$\text{PoI}(\mathbf{x}) = P(Y(\mathbf{x}) \leq f_{min}) = \int_{-\infty}^{f_{min}} \phi(Y(\mathbf{x})) dY = \Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) \quad (2)$$

$$\text{ExI}(\mathbf{x}) = \int_{-\infty}^{f_{min}} (f_{min} - y) \phi\left(\frac{y - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) dy \quad (3)$$

$$= (f_{min} - \hat{y}(\mathbf{x})) \Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x}) \phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) \quad (4)$$

In this, assume we are using a Kriging model to minimize a function $f(\mathbf{x})$ with the best known function value f_{min} . Here, $\hat{y}(\mathbf{x})$ is the approximated value with the corresponding standard error $\hat{s}(\mathbf{x})$, and for LB is w a user-defined weight. Furthermore, $\phi(\cdot)$ is the probability density function of the normal distribution, and $\Phi(\cdot)$ is the cumulative distribution function.

Over the years, several papers have been published that extend these prescreening functions. To control the balance between global and local search, Schonlau et al. [16] introduced the g parameter and the Generalized Expected Improvement (GEI). Following this, Sasena et al. [17] suggested to use an annealing scheme to control the g parameter of the GEI prescreening function. Authors have also suggested to parallelize the use of ExI. For example, Janusevskis et al. [18] suggested the q-EI as a way to generate multiple points to

evaluate in each optimization run thereby lowering the number of runs. In a similar direction, Ponweiser et al. [19] introduced the MGEI and the CMGEI criteria and compared these with the annealing of Sasena et al. [17]. In this comparison, GEI turned out to be the best approach. Although the parallelization of ExI is interesting, saving optimization run-time is less relevant as 90-95% of the computation time is typically spent on the actual evaluation of new solutions.

At this point, it may be worthwhile to take a step back and study the functions from a more mathematical perspective. In the following, assume (WLOG) that $f_{min} = 0.0$ and that both function values and standard errors have been normalized. Thus, a negative \hat{y} corresponds to a value that is better than the best known solution. Figure 1 displays the surface plot of PoI and ExI for different values of \hat{y} and \hat{s} . The plot for LB is omitted as it resembles the plot for ExI. The contour lines illustrate solutions that are considered equal by the plotted prescreening function.

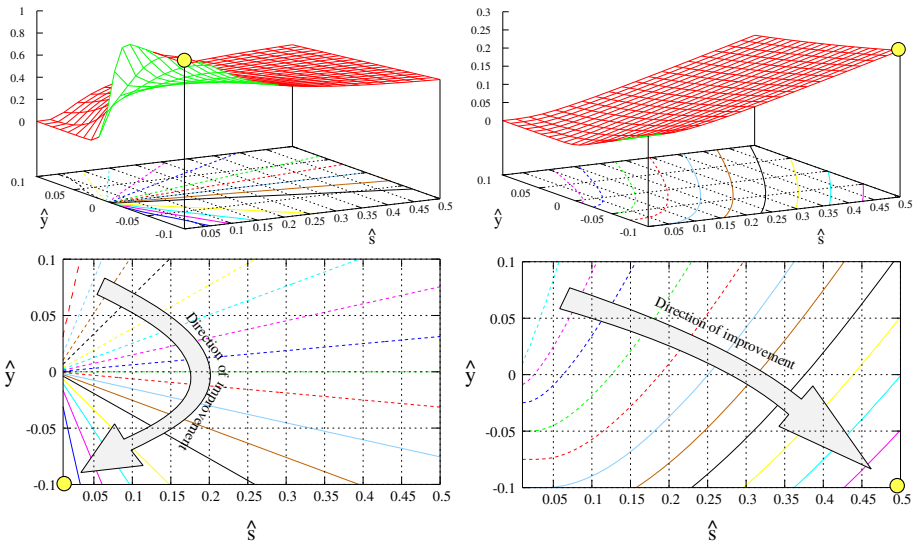


Fig. 1. Surface and contour plots of PoI (left) and ExI (right). Yellow circle is max (in the given range).

As seen in the figure, PoI has maximum when (or rather if) the optimization algorithm achieves a solution that outcompetes the current best known, i.e., $\hat{y} < 0$, and the standard deviation \hat{s} is small. In case the evaluated solution is worse than the best known, i.e., $\hat{y} > 0$, then PoI favors solutions with large standard deviation \hat{s} . Furthermore, the PoI function is 0.5 for all values of \hat{s} when $\hat{y} = 0$. In contrast to PoI, the ExI function clearly favors solutions with the largest possible \hat{s} regardless of the approximated mean \hat{y} .

In an investment perspective, PoI implements a *low risk strategy* as it promotes a low standard deviation when a solution with a potential improvement

($\hat{y} < 0$) is found. For example, given two solutions x_1 and x_2 both with $\hat{y} = -0.05$ then PoI will favor the solution with lowest \hat{s} . In contrast, ExI represents a *high risk strategy* as ExI (and also LB) prefer unknown points that maximize the standard deviation. Considering financial investment as another field involving risk strategies, one general recommendation is to employ a *risk spreading strategy*. Thus, the strategies implemented by LB, PoI, and ExI are in conflict with this general recommendation.

2.4 Investment Portfolio Improvement Prescreening

The main idea in investment portfolio improvement (IPI) prescreening is to optimize toward a target standard deviation t . This idea can be represented by numerous IPI functions. In this study, a number of functions were tested in preliminary runs. The IPI function defined in equation 5 turned out to have the best performance on the standard benchmark functions introduced in section 3.

$$IPI(\mathbf{x}) = 0.5 \cdot \Phi\left(\frac{f_{min} - \hat{y}}{1.05 - t}\right) + \Phi\left(\frac{-(\hat{s} - t)^2}{0.05}\right) \tag{5}$$

Figure 2 shows the function for $t = 0.4$ and $t = 0.8$. For a low t , the function promotes a search toward local improvements of the best known point. For a high t , the function primarily induces a search for a solution with the desired standard deviation and secondly an improvement over best known solution. The functions studied in preliminary runs did not impose a sufficiently strong selection pressure for improvement, i.e., the found solutions had the desired standard deviation, but with suboptimal performance.

To perform actual *portfolio optimization*, the algorithm needs to search with multiple values for t simultaneously. Naturally, this can be done in numerous ways. For instance, one may use an island model [20], a cellular EA [20], a multinational model [21], or other variants of diversity maintaining techniques. In this study, the target values t are set from the individual's population index i in the differential evolution algorithm. Thus, the IPI function for individual i at iteration j is defined as in equation 6.

$$IPI(\mathbf{x}_i, j) = 0.5 \cdot \Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x}_i)}{1.05 - t_i}\right) + \Phi\left(\frac{-(\hat{s}(\mathbf{x}_i)/n_f - t_i)^2}{0.05}\right) \tag{6}$$

$$t_i = \frac{i}{popsize - 1} \quad (\text{target for individual } i) \tag{7}$$

$$n_f = \max_i(\hat{s}(p_{i,j-1})) \quad (\text{normalization factor}) \tag{8}$$

In this, the calculation of the normalization factor is based on measurements from the previous generation $j - 1$ where $p_{i,j-1}$ denotes parent i . Optimizing under this scheme results in a population where the low-index individuals seek a low standard deviation and high-index individuals have a high standard deviation.

In a real-world application, the resources for evaluating found solutions are often limited by the time and costs required to perform the full evaluation.

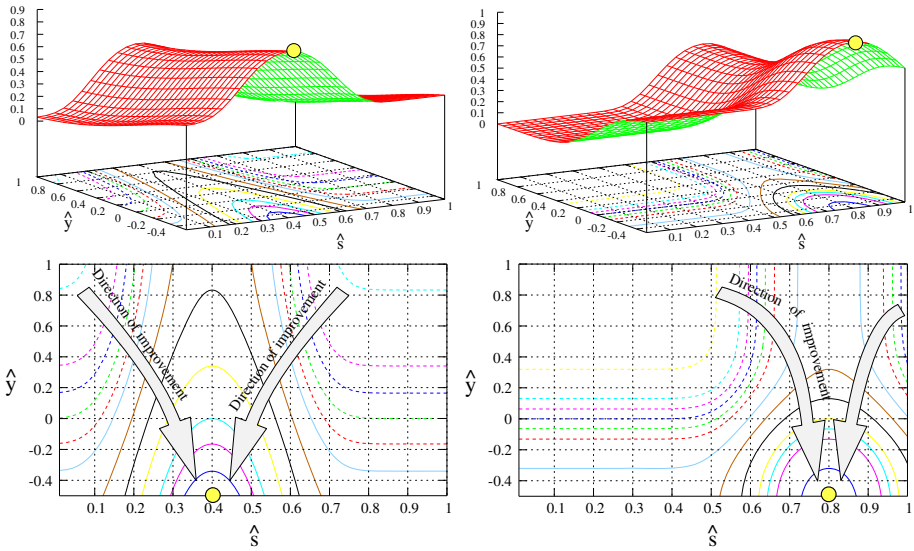


Fig. 2. Surface and contour plots for IPI with $t = 0.4$ and $t = 0.8$. Yellow circle is max.

The optimization specialist is therefore often only interested in 3-5 distinct candidate solutions per round [22]. The IPI algorithm therefore returns a user-defined number of candidates K_{NC} as follows. After the stopping criterion is met, the algorithm divides the population into K_{NC} segments and return the highest IPI scoring individual from each segment. For example, setting $K_{NC} = 3$ will result in a low-risk, a mid-risk, and a high-risk solution taken from the first third, the middle third, and the last third of the population.

3 Experimental Setup

The experiments focus on comparing the performance of the novel IPI algorithm with the established PoI and ExI prescreening functions, and a DE version of the annealing GEI algorithm [17]. To complete the picture, the DE algorithm is also optimizing only the mean value (OMV), thereby allowing comparison with the traditional approach. The GEI prescreening function is defined according to equation 9 and g values are set from table 1 as in [17].

$$E(I^0) = P(y < f_{min}) = \Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) = \Phi(u) \quad g = 0 \tag{9}$$

$$E(I^g) = \hat{s}(\mathbf{x})^g \sum_{k=0}^g (-1)^k \binom{g}{k} u^{g-k} T_k \quad g = 1, 2, \dots$$

$$T_0 = \Phi(u), \quad T_1 = -\Phi(u), \quad T_k = -u^{k-1} \phi(u) + (k-1)T_{k-2} \quad k = 2, 3, \dots$$

Table 1. Annealing values for g depending on number of iterations

Iteration	1-4	5-9	10-19	20-24	25-34	≥ 35
g	20	10	5	2	1	0

The DE algorithm was run with 100 individuals, $CR = 0.2$, $F = 0.35$, and 1000 iterations to ensure convergence. In the IPI algorithm, the number of candidates was set to $K_{NC} = 3$ as this represent a typical number of simulations that can be performed per day for an industrial problem. Thus, the algorithm returned a low-risk, a medium-risk, and a high-risk solution.

The test suite includes seven benchmark problems and two model-calculated pump design problems. The tested benchmark problems are the well-known¹ Branin function, the six hump camel back function, the Hartmann 3D function, the Hartmann 6D function, the Colville function, the Rastrigin 2D function, and the less known Sasena “mystery” function [23], which is defined in equation 10.

$$\begin{aligned} \min f(x_1, x_2) = & 2 + 0.01(x_2 - x_1^2)^2 + (1 - x_1)^2 + 2(2 - x_2)^2 + \quad (10) \\ & 7 \sin(0.5x_1) \cdot \sin(0.7x_1x_2) \\ & x_i \in [0 : 5], i = 1, 2 \end{aligned}$$

The benchmark problems are chosen to represent engineering-like problems that typically have a few local optima and a single global optimum. However, the Rastrigin function does not fulfill this selection criterion as it has 11 optima per dimension (Rastrigin 2D has 120 local optima and one global). Nevertheless, it is included to investigate the performance on a simple problem often used in traditional tests of optimization algorithms.

The two model-calculated pump design problems are based on classic pump textbook theory [24] coupled with in-house loss models for modeling the Grundfos pumps. The details of the two pump design problems cannot be revealed as it would violate the need for business confidentiality. However, the first problem has 6 design parameters and the objective is to maximize the hydraulic efficiency in the design point. The second problem has 12 design parameters and the objective is also to maximize the hydraulic efficiency in the design point.

All nine problems are fast to calculate and allow a statistical comparison of the methods based on 20 repetitions each executed as follows:

1. Generate 20 random solutions and evaluate them.
2. While (Total number of new solutions ≤ 50)
 - (a) Train Kriging approximator on database.
 - (b) Run the DE algorithm with the prescreening function.
 - (c) Add 1-3 new solution(s) to database (IPI adds 3, others add 1).
3. Report the best found solution.

¹ Details are omitted due to space limitations.

The initial database of 20 random solutions and the following 50 samples represents a typical setup in the industry. For example, a computational fluid dynamics (CFD) simulation of a full pump curve can take up to 2-3 hours in a steady-state setup and up to 4-5 days for a full transient simulation. The number of initial solutions was deliberately kept at 20 individuals to stress the algorithms as the problem dimensionality increased.

4 Results and Discussion

The results of the experiments are listed in table 2. In the table, a number marked in **bold** denote the algorithm with the best mean. Furthermore, a dagger (†) indicates that the algorithm is best wrt. Mann-Whitney rank sum test, i.e., the null-hypothesis² H_0 is rejected at 5% confidence level and a double dagger (‡) at 1% confidence level.

Table 2. Mean and standard deviation for the seven benchmark problems and the two pump problems

Function	OMV	PoI	ExI	GEI	IPI
Branin 2D	2.57±1.795	2.95±2.371	0.41±0.017	0.42±0.019	0.40±0.003‡
Sasena 2D	1.29±1.780	1.51±2.297	-1.23±0.965	-1.41±0.117	-1.45±0.022
Six hump 2D	0.73±3.387	0.00±0.907	-0.90±0.110	-0.75±0.209	-0.92±0.116
Rastrigin2D	10.36±5.256	11.23±4.413	2.69±2.440	7.14±5.173	3.15±2.470
Hartmann 3D	-2.96±0.486	-3.26±0.482	-3.77±0.106	-3.71±0.170	-3.82±0.056†
Colville 4D	8519±17993	4480±6256	749±1153	472±620	829±1108
Hartmann 6D	-1.39±0.500	-1.23±0.576	-1.72±0.660	-1.99±0.707	-2.77±0.472†
Pump 6D	45.97±3.137	46.54±2.407	49.39±1.484	48.85±1.167	49.53±1.205
Pump 12D	59.57±1.806	59.57±1.477	61.28±1.133	60.52±1.243	61.34±1.690

As seen, OMV and PoI clearly have the worst performance on all problems. Comparing ExI, GEI, and IPI, the IPI achieves a better mean on seven of the nine tested problems and three of these seven are further supported by the Mann-Whitney rank sum tests. Interestingly, the GEI does not seem to be significantly better than traditional ExI. One possible explanation is that a g value higher than 1 actually induces an even stronger focus on finding solutions with high standard deviation. Stepping from traditional ExI ($g = 1$) to PoI ($g = 0$) occurs rather late in the annealing process and this step represents a rather large change in search strategy, i.e., from high-risk to low-risk as discussed earlier. Thus, the annealing approach could probably have benefited from smaller steps in the g value from, e.g., 1.0 to 0.9, and gradually toward 0.0. However, this is not possible with the current formulation of GEI.

Scrutinizing the IPI data, a typical run benefits from the portfolio optimization as follows (recall that $K_{NC} = 3$ and the algorithm adds 50 new solutions).

² The null-hypothesis states that the samples are drawn from the same distribution.

In the beginning of the run, the medium or high-risk solutions often locate a new best point, which are further improved by the low risk solution in following rounds. Towards the end, the high-risk solutions often explore regions with sub-optimal performance as these parts have not yet been explored. In a few runs, the high-risk solution discovers a new best point towards the end of the run. Thus, the algorithm clearly benefits from implementing the portfolio strategy.

5 Conclusions

This paper presents the investment portfolio improvement prescreening approach and demonstrates its performance on seven benchmark problems and two real-world pump design problems. The experiments show that the suggested technique yields a better mean value in seven of the nine tested functions. Thus, the experiments support the initial analysis of the search strategies employed by established prescreening functions like Probability of Improvement (low risk) and Expected Improvement (high risk). Hence, it is an advantage to use a risk-spreading strategy as this simultaneously allows search in the vicinity of the best known solution and explorative search in new regions of the search space.

Regarding future work, this initial paper only presents the main idea and demonstrates its potential. The scheme is easy to extend and some next steps could be to investigate an annealing scheme where the target standard deviation t is gradually lowered or alternatively a self-adaptive version. Another idea is to implement an island version of the method.

References

1. Dennis, J., Torczon, V.: Managing approximate models in optimization. In: Alexandrov, N., Hussani, M. (eds.) *Multidisciplinary design optimization: State-of-the-art*, pp. 330–347. SIAM (1997)
2. Ulmer, H., Streichert, F., Zell, A.: Evolution strategies assisted by gaussian processes with improved pre-selection criterion. In: Sarker, R., Reynolds, R., Abbass, H., Tan, K.C., McKay, B., Essam, D., Gedeon, T. (eds.) *Proceedings of the 2003 Congress on Evolutionary Computation CEC 2003*, December 8–12, pp. 692–699. IEEE Press, Canberra (2003)
3. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13(4), 455–492 (1998)
4. Sheng, N., Liao, C., Lin, W., Chang, L., Zhang, Q., Zhou, H.: A hybrid optimized algorithm based on ego and taguchi's method for solving expensive evaluation problems of antenna design. *Progress In Electromagnetics Research C* 17, 181–192 (2010)
5. Jeong, S., Minemura, Y., Obayashi, S.: Optimization of combustion chamber for diesel engine using kriging model. *Journal of Fluid Science and Technology* 1(2), 138–146 (2006)
6. Couckuyt, I., Declercq, F., Dhaene, T., Rogier, H., Knockaert, L.: Surrogate-based infill optimization applied to electromagnetic problems. *International Journal of RF and Microwave Computer-Aided Engineering* 20(5), 492–501 (2010)

7. Emmerich, M.T., Giannakoglou, K.C., Naujoks, B.: Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. *IEEE Transactions on Evolutionary Computation* 10(4), 421–439 (2006)
8. Das, S., Mukherjee, R., Kundu, R., Vasilakos, T.: Multi-user detection in multi-carrier cdma wireless broadband system using a binary adaptive differential evolution algorithm. In: Blum, C., Alba, E., Auger, A., Bacardit, J., Bongard, J., Branke, J., Bredeche, N., Brockhoff, D., Chicano, F., Dorin, A., Doursat, R., Ekart, A., Friedrich, T., Giacobini, M., Harman, M., Iba, H., Igel, C., Jansen, T., Kovacs, T., Kowaliw, T., Lopez-Ibanez, M., Lozano, J.A., Luque, G., McCall, J., Moraglio, A., Motsinger-Reif, A., Neumann, F., Ochoa, G., Olague, G., Ong, Y.S., Palmer, M.E., Pappa, G.L., Parsopoulos, K.E., Schmickl, T., Smith, S.L., Solnon, C., Stuetzle, T., Talbi, E.G., Tauritz, D., Vanneschi, L. (eds.) *GECCO 2013: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, Amsterdam, The Netherlands, July 6-10, pp. 1245–1252. ACM (2013)
9. Ursem, R.K., Vadstrup, P.: Parameter identification of induction motors using differential evolution. In: *Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003)*, pp. 790–796 (2003)
10. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkley (1995)
11. Storn, R., Price, K.: Differential evolution a simple and efficient heuristic for global optimisation over continuous spaces. *Journal of Global Optimization* 11, 341–359 (1997)
12. Martin, J.D., Simpson, T.W.: Use of kriging models to approximate deterministic computer models. *AIAA Journal* 43(4), 853–863 (2005)
13. Laurenceau, J., Sagaut, P.: Building efficient response surfaces of aerodynamic functions with kriging and cokriging. *AIAA Journal* 46(2), 498–507 (2008)
14. Matheron, G.: Principles of geostatistics. *Economic Geology* 4(4), 409–435 (1963)
15. Lophaven, S.N., Nielsen, H.B., Søndergaard, J.: Dace – a matlab kriging toolbox. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark (2002)
16. Schonlau, M., Welch, W.J., Jones, D.R.: Global versus local search in constrained optimization of computer models. In: *New Developments and Applications in Experimental Design. IMS Lecture Notes - Monograph Series*, vol. 34, pp. 11–25 (1998)
17. Sasena, M.J., Papalambros, P., Goovaerts, P.: Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization* 34, 263–278 (2002)
18. Janusevskis, J., Le Riche, R., Ginsbourger, D., Girdziusas, R.: Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges. In: Hamadi, Y., Schoenauer, M. (eds.) *LION 2012. LNCS*, vol. 7219, pp. 413–418. Springer, Heidelberg (2012)
19. Ponweiser, W., Wagner, T., Vincze, M.: Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models. In: Wang, J. (ed.) *2008 IEEE World Congress on Computational Intelligence*, Hong Kong, June 1-6. *IEEE Computational Intelligence Society*, IEEE Press (2008)
20. Bäck, T., Fogel, D.B., Michalewicz, Z., et al. (eds.): *Handbook on Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press (1997)
21. Ursem, R.K.: Multinational evolutionary algorithms. In: Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzal, A. (eds.) *Proceedings of the Congress of Evolutionary Computation (CEC 1999)*, Mayflower Hotel, Washington D.C., USA, July 6-9, vol. 3. *IEEE Press* (1999)

22. Ursem, R.K., Justesen, P.D.: Multi-objective distinct candidates optimization: Locating a few highly different solutions in a circuit component sizing problem. *Applied Soft Computing* 12(1), 255–265 (2012)
23. Sasena, M.J.: Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. PhD thesis, University of Michigan (2002)
24. Gülich, J.: *Centrifugal Pumps*. Springer (2010)

Distance Measures for Permutations in Combinatorial Efficient Global Optimization

Martin Zaefferer, Jörg Stork, and Thomas Bartz-Beielstein

Cologne University of Applied Sciences
Faculty for Computer and Engineering Sciences, 51643 Gummertsbach, Germany
`firstname.lastname@fh-koeln.de`

Abstract. For expensive black-box optimization problems, surrogate-model based approaches like Efficient Global Optimization are frequently used in continuous optimization. Their main advantage is the reduction of function evaluations by exploiting cheaper, data-driven models of the actual target function. The utilization of such methods in combinatorial or mixed search spaces is less common. Efficient Global Optimization and related methods were recently extended to such spaces, by replacing continuous distance (or similarity) measures with measures suited for the respective problem representations.

This article investigates a large set of distance measures for their applicability to various permutation problems. The main purpose is to identify, how a distance measure can be chosen, either a-priori or online. In detail, we show that the choice of distance measure can be integrated into the Maximum Likelihood Estimation process of the underlying Kriging model. This approach has robust, good performance, thus providing a very nice tool towards selection of a distance measure.

1 Introduction

One frequent issue in real-world optimization problems are costly objective function evaluations. These may be caused by time-consuming simulations or complex trials and measurements. In continuous optimization, surrogate-model based approaches use cheaper, data-driven models to reduce the number of objective function evaluations, e.g. in the Efficient Global Optimization (EGO) algorithm [16]. In combinatorial optimization, surrogate models received less attention. Recently, approaches from continuous modeling and optimization have been extended to mixed or purely combinatorial problem spaces: Radial Basis Function Networks (RBFN), Kriging, and EGO [22,34]. A short overview of these previous studies will be given in Sec. 2.

The employed modeling tools base their prediction on measures of similarity or distance between candidate solutions. The core idea of the extension is therefore to replace the distance measures used in continuous spaces (e.g., Euclidean) with distance measures more suited for the given problem representation. Two questions arise in this context: First, which distance measure is most suited? And second, how can this measure be chosen a priori as well as during the optimization procedure for a given problem?

This article tries to provide answers to both questions for an important solution representation type: permutations. The permutation representation is required in a large array of problems [2]. The reader may consider production processes, which have to be divided into several jobs to be scheduled for one or more machines in order to achieve a timely completion. Here, several distance measures will be used to handle various problem classes and instances. The employed distance measures, Kriging and EGO will be introduced in Sec. 3. Their performance will be examined in an experimental study, as outlined in Sec. 4. Observations will be described and discussed in Sec. 5. Finally, a summary and an outlook on future research are given in Sec. 6.

2 Previous Research

Compared to their frequent use for continuous problem domains, surrogate model driven approaches are relatively unknown in combinatorial or mixed optimization [15]. Voutchkov et al. [31] introduce an expensive optimization problem for signed permutations, concerning weld sequence optimization. Regarding data-driven approaches for black-box problems (which are in the focus of this paper), Li et al. [21] proposed Radial Basis Function Network (RBFN) models based on a weighted distance measure, replacing the usual distance measure employed in RBFN. Their RBFN models were able to model to mixed-integer problems. Mixed problems also occur in algorithm tuning, where continuous, discrete, and categorical parameters may occur. In this context, Random Forest models have been used due to their ability to capture discrete and categorical parameters [4]. Hutter [14] also describes a Kriging model with a Hamming distance based kernel function to handle categorical variables.

Moraglio and Kattan [22] adapted an RBFN to arbitrary distance measures to model arbitrary combinatorial optimization problems. Their approach has also been applied to Quadratic Assignment Problems (QAP) [23]. The same conceptual extension for Kriging was recently investigated by Zaefferer et al. [34]. This allowed to apply the Kriging based Efficient Global Optimization (EGO) algorithm to combinatorial problems. Kriging-based EGO performed very well, in comparison to other model-driven or model-free approaches. Furthermore, the choice of distance measure was shown to have a very strong influence on optimization performance.

In this article, we will focus on Kriging-based EGO only. We will look at a much larger array of permutation problems and distance measures. Our goal is to derive recommendations for the selection of problem-specific distance measures.

3 Methods

3.1 Distance Measures

Previously, distance measures for permutations were investigated, e.g., for the purpose of landscape analysis [27] or diversity preservation [29]. These previous studies illustrate that a large array of distance measures is available. For the

Table 1. Investigated distance measures. Second column lists runtime complexity. Third column lists median runtime of 1000 evaluations for permutations of length 30.

Name	complexity	runtime [μs]	Abbrev.
Levenshtein	$O(n^2)$	7	Lev
Swap	$O(n^2)$	6	Swa.
Interchange	$O(n^2)$	14	Int.
Longest Common Subsequence	$O(n^2)$	8	LCSeq
Longest Common Substring	$O(n^2)$	8	LCStr
R	$O(n^2)$	5	R
Adjacency	$O(n^2)$	6	Adj.
Position	$O(n^2)$	6	Pos.
Position ²	$O(n^2)$	6	Posq.
Hamming	$O(n)$	2	Ham.
Euclidean	$O(n)$	6	Euc.
Manhattan	$O(n)$	4	Man.
Chebyshev	$O(n)$	3	Che.
Lee	$O(n)$	6	Lee

purpose of distance-based modeling, only Hamming, Swap and Interchange distance [23,34] were used. In this study, we will analyze 14 different distance measures, as summarized in Table 1. The given runtime complexity refers to the employed implementations. More efficient variants may be available. To avoid scaling bias, all distance measures are scaled to yield values from [0; 1].

In the following, we describe basic features of these distance measures. Since naming of measures in literature varies, this clarification is useful to avoid confusion.

- Levenshtein and Edit distance are sometimes used as synonyms. In fact, Levenshtein is only one example of an edit distance. It counts the minimum number of deletions, insertions, or substitutions required to transform one string (or here: permutation) into another. For an implementation we refer to [32].
- A swap operation is the transposition of two adjacent elements in a permutation. The Swap distance is defined as the minimum number of swaps required to transform one permutation into another. It has also been called Precedence distance [27], or Kendall’s Tau [19,29]. For permutations of length n it is [29]:

$$\delta_{Swa.}(\pi, \pi') = \sum_{i=1}^n \sum_{j=1}^n z_{ij} \quad \text{where} \quad z_{ij} = \begin{cases} 1 & \text{if } \pi_i < \pi_j \text{ and } \pi'_i > \pi'_j, \\ 0 & \text{otherwise.} \end{cases}$$

- An interchange operation is the transposition of two arbitrary elements. Respectively, the Interchange (also: Cayley) distance is the minimum number of interchanges required to transform one permutation to another [27].
- The longest common subsequence distance counts the largest number of elements that follow each other in both permutations, with interruptions. We use the algorithm described in [13].
- The longest common-substring distance counts the largest number of elements that follow each other in both permutations, without interruption, i.e., all elements are adjacent. We use the implementation in [33].

- The R-distance [9,29] counts the number of times that one element follows another in one permutation, but not in the other. It is identical with the uni-directional adjacency distance [25]. It is computed by

$$\delta_R(\pi, \pi') = \sum_{i=1}^{n-1} y_i \text{ where } y_i = \begin{cases} 1 & \text{if } \exists j : \pi_i = \pi'_j \text{ and } \pi_{i+1} = \pi'_{j+1}, \\ 0 & \text{otherwise.} \end{cases}$$

- The (bi-directional) adjacency distance [25,27] counts the number of times two elements are neighbors in one, but not in the other permutation. Unlike R-distance (uni-directional), the order of the two elements does not matter.
- The Position distance [27] is identical with the Deviation distance or Spearman’s footrule [29],

$$\delta_{Pos}(\pi, \pi') = \sum_{k=1}^n |i - j| \text{ where } \pi_i = \pi'_j = k .$$

- The Squared Position distance is Spearman’s rank correlation coefficient [29]. In contrast to the Position distance, the term $|i - j|$ is replaced by $(i - j)^2$
- The Hamming distance or Exact Match distance simply counts the number of unequal elements in two permutations, i.e.,

$$\delta_{Ham.}(\pi, \pi') = \sum_{i=1}^n a_i \text{ where } a_i = \begin{cases} 0 & \text{if } \pi_i = \pi'_i, \\ 1 & \text{otherwise.} \end{cases}$$

- The Euclidean distance is

$$\delta_{Euc.}(\pi, \pi') = \sqrt{\sum_{i=1}^n (\pi_i - \pi'_i)^2} .$$

- The Manhattan distance (A-Distance [29,9]) is

$$\delta_{Man.}(\pi, \pi') = \sum_{i=1}^n |\pi_i - \pi'_i| .$$

- The Chebyshev distance is

$$\delta_{Che.}(\pi, \pi') = \max_{1 \leq i \leq n} (|\pi_i - \pi'_i|) .$$

- The Lee distance [20] can be adapted to permutations with

$$\delta_{Lee}(\pi, \pi') = \sum_{i=1}^n \min(|\pi_i - \pi'_i|, n - |\pi_i - \pi'_i|) .$$

The reversal distance (number of reversals required to transform one permutation to another) was not used, even though it is especially promising for the Traveling Salesperson Problem (TSP). Calculating the reversal distance for unsigned permutations is NP-hard [10].

3.2 Kriging for Combinatorial Optimization

Kriging is a very flexible predictor, that models the correlation between samples, assuming that they are derived from a Gaussian process. Kriging also provides an estimate of uncertainty of its own prediction. For a detailed description of Kriging, we refer to Forrester et al. [11]. The adaptation to combinatorial or mixed problems was described by Zaefferer et al. [34].

Training a Kriging model requires the set of m solutions $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1\dots m}$ with observations $\mathbf{y} = \{y^{(i)}\}_{i=1\dots m}$. The predicted mean of a new candidate solution \mathbf{x} is referred to as $\hat{y}(\mathbf{x})$, the estimated error of that prediction is $\hat{s}(\mathbf{x})$. The Kriging model has the parameters θ , p , $\hat{\sigma}$, and $\hat{\mu}$. Maximum Likelihood Estimation (MLE) is used to determine these parameters and requires a matrix inversion. In [34], standard inversion was used. We observed a problem with

standard inversion for 4 out of the 14 distance measures (Int., Lev., LCSeq, Che.). While all others worked well, these four measures may produce numerical instability. Hence, the more stable inversion via Cholesky decomposition is used, which requires a positive semi-definite correlation matrix.

3.3 Choosing a Distance Measure in Kriging

In standard Kriging, the distance measure is not fixed. Rather, it can partially be understood as a parametrized distance measure. E.g., it may resemble Euclidean ($p = 2$) or Manhattan ($p = 1$) distance.

The choice of distance measure can also be understood as a (categorical) parameter of the model. Hence, we suggest to perform MLE for each distance measure separately. Afterwards, the distance measure with maximum likelihood is chosen for the model. This procedure repeats every time the model is build, i.e., in each iteration of a single EGO run. In the experimental study this will be referred to as “All”.

A wrong decision may occur, especially while data is still very sparse. Therefore, we expect the performance of choosing a distance measure with MLE to be equal to or worse than the best single measure. An exception would be the case where the underlying optimization problem has a dynamic behavior. Then, different measures may be preferable in different phases of the optimization run.

3.4 Efficient Global Optimization

EGO was introduced by Jones et al. [16]. In this algorithm, a Kriging model is first build based on an initial set of solutions. If the uncertainty $\hat{s}(\mathbf{x}) > 0$, one can compute the Expected Improvement (EI) of a candidate solution, otherwise $EI(\mathbf{x}) = 0$. EI determines how much improvement can be expected from the candidate solution, and thus balances exploitation vs. exploration. The solution that maximizes EI is evaluated with the target function. The result is used to update the Kriging model. This is repeated until a termination criterion (e.g., function evaluation budget) is fulfilled.

4 Experimental Setup

4.1 Correlation between Distances

As a first step, correlation between the 14 different distance measures is investigated. Distances between all permutations of length $n = 7$ are computed (i.e., 5040 distance values for each measure), and the correlation is calculated.

4.2 Matrix Condition

To quickly assess whether all measures yield positive semi-definite correlation matrices (as required for MLE), we performed an experimental test. Ten solutions were created randomly, while another 90 were created by consecutive interchange mutations. This yields 100 solutions of varying distances. This was done for various permutation lengths ($n = \{5, 6, 7, 8, 9, 10, 20, 50, 100\}$). In case of the smallest instance, the 100 solutions represent a very large section of the search space (which has a size of $n! = 120$). Larger instances are less crowded.

Since θ will also influence the correlation matrix condition, it was varied from 10^{-10} to 10^{10} . For each distance measure, each dimension n , and each θ the correlation matrix is computed and its condition checked.

4.3 Benchmark Problems

For all further experiments, five permutation problem classes are used.

- As in [34], four instances of the Quadratic Assignment Problem (QAP) [6] from the QAPLIB [7] are chosen (`nug30`, `nug12`, `tho30` and `kra32`). In the QAP n facilities have to be assigned to n locations. Assignment cost is minimized, based on flow between facilities and distance between locations.
- Four instances of the Flow-shop Scheduling Problem (FSP) [30] are chosen (`reC05`, `reC13`, `reC19`, `reC31` [24]) from the OR-Library [5]. Here, the finishing time of the last of n jobs sequenced on m machines is minimized.
- Three TSP instances are chosen from the TSPLIB [26] (`bayg29`, `fri26`, `gr24`). In the TSP, the cost or length of a route through several locations is minimized. Each location has to be visited once.
- Three instances of the Asymmetric TSP (ATSP) are generated (`atsp10`, `atsp20`, `atsp30`). For each instance, a distance matrix is created randomly with a uniform distribution. The three instances are of size 10, 20, and 30. In contrast to TSP, the cost of traveling between two locations depends on direction.
- Finally, four instances of the single-machine total Weighted Tardiness problem (WT) [1] are chosen, also from the OR-Library [5] (the first four of length 40, i.e., `wt40a`, `wt40b`, `wt40c`, `wt40d`). Here, n jobs are sequenced on one machine that can handle one job at a time. The tardiness of a schedule for all jobs, weighted by a set of n given weights is minimized. It depends on the given processing times and due dates of each job.

For QAP, TSP, ATSP and WT the length of the permutation n is given by the number in the instance name. For FSP, n is 20, 20, 30 and 50 for `reC05`, `reC13`, `reC19` and `reC31` respectively.

We use this benchmark set under the artificial assumption of costly target function evaluation. While some of these problems have actual real world relevance (e.g., based on real world data), none may be considered expensive. This allows for a more in-depth study, providing first results, which of course should be validated with actually expensive problems in future studies.

4.4 Local Fitness Distance Correlation

Fitness Distance Correlation [17] is a measure for the analysis of fitness landscapes. It measures correlation between fitness value and distance to the known global optimum. When the optimum is unknown, it can be replaced by the best solution in the set, thus yielding the Local FDC (LFDC) [18]. Here, LFDC will be calculated based on 20,000 unique, randomly created individuals for each instance. When minimizing an easy problem, positive correlation is expected to occur. Thus, the LFDC values may represent an indicator of problem difficulty. We are interested in LFDC from a different perspective. It is investigated, whether LFDC can be used to identify a suitable distance measure for a given

problem. LFDC may be unsuited towards this end, as previous studies already showed that FDC may be misleading for certain problems [3].

4.5 Optimization Performance

Finally, we compare optimization performance. To that end, EGO, a model-free Genetic Algorithm (GA), Random Search (RS) and a simple 2-opt local search are employed with a strictly limited budget of 200 function evaluations. GA, RS and 2-opt are baselines in this comparison. Their main purpose is to identify whether the various EGO variants work at all.

The GA used in the comparison will use cycle crossover and the mutation operator is an interchange of arbitrary elements. Furthermore the algorithm will use a population size of ten, crossover rate 0.5, mutation rate $1/n$, tournament selection with tournament size two and tournament probability 0.9. The EGO algorithm will start with an initial set of ten solutions. Kriging parameter p is set to one, while the others are determined with MLE. Internally, EGO will perform optimization of the (assumed to be cheap) surrogate model. Hence, the same GA is used with 10,000 model evaluations, and a population size of 20.

For a fair comparison of actual competitors, most of the mentioned parameters would require tuning. Since the basic GA is just a baseline, this is not necessary. The EGO variants use identical settings thus yielding a fair comparison among themselves.

5 Observations and Discussion

5.1 Correlation and LFDC

Correlation between distance measures and LFDC values are depicted in Fig. 1. Measures that correlate also have comparable LFDC (e.g., Pos. and Posq.). The LFDC for QAP, TSP & ATSP as well as FSP & WT have similar structure.

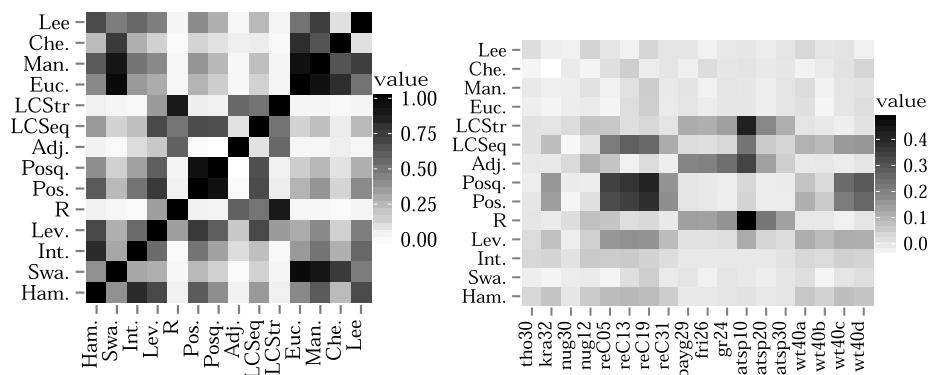


Fig. 1. Heatmaps of distance measure correlation (left), and LFDC values (right)

5.2 Matrix Condition

For most distance measures, a positive semi-definite matrix could be determined for each n . Only Adjacency distance with $n = 5$ and $n = 6$ did not yield any positive semi-definite matrices. Hence, one should avoid using Adjacency distance when the training samples represent a large portion of the search space.

5.3 Optimization Performance

Figure 2 shows the results of the optimization experiments. Each EGO variant is referred to by the name of the employed distance measure. Results of 2-opt are not shown, for the sake of brevity. 2-opt usually ranks worse than GA and only outperforms GA for the three TSP instances. Still, it can not compete with the model-based approaches. Chebyshev distance (Che.) and RS are consistently outperformed by the GA and hence not included in the plot.

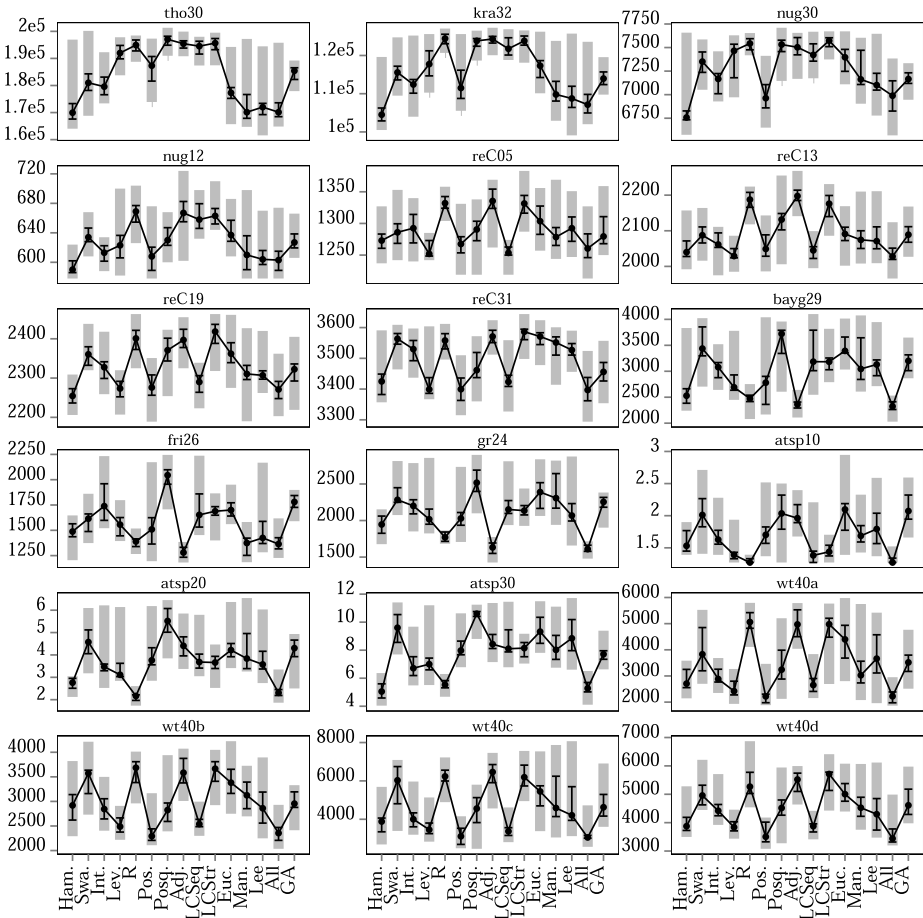


Fig. 2. Optimization performance: Dots are median, black bars are interquartile range, thick grey bars are range from minimum to maximum. Smaller values are better.

Three main groups with similar structure can be identified: first, the QAP instances, second, the TSP and ATSP instances, and third, the WT and FSP instances. Members of each group have a similar pattern, although the best performing method may not be identical for all members. These three blocks do coincide with the structure that is visible in the LFDC results. LFDC may identify the best measure (see e.g., R-Distance and atsp10) or may fail completely (e.g., Posq. and reC19). LFDC is apparently unsuited to identify a proper distance measure.

Overall, the model-free GA is always outperformed by at least 5 EGO variants. Choosing the wrong distance measure may however lead to performance worse than the model-free GA. Choosing a distance measure with MLE (*All*) never ranks worse than 3rd best, making it the most robust method in this test bed. *All* ranks first place in 7 of 18 instances. It seems that the underlying problem does profit from a dynamic choice of distance measure. This behavior may be related to dynamic behavior observed for the choice of mutation operators through self-adaption [28]. Many distance measures can be related to specific mutation operators. The single best distance measure is Hamming distance, yielding best results in 6 of the 18 test problems, but receiving lower ranks for other instances. For each problem class, the single best distance measures are: Ham. for QAP, Lev. for FSP, Adj. for TSP, R for ATSP, Pos. for WT. While the scheduling problems rather reflect the importance of a relative order, TSP or ATSP are more concerned with adjacency of neighboring cities. Hence, it makes sense that a bi-directional adjacency measure is used for TSP, while uni-directional adjacency (R-Distance) is used for the ATSP instances. In ATSP, direction matters, whereas in TSP it does not.

The nice and robust performance of choosing a distance measure with MLE makes for a promising result. Here, the only issue is to carefully avoid numerical problems, i.e., to use matrix inversion via Cholesky decomposition. Should the increased computational effort necessitate a smaller set of distance measures, Hamming distance should always be included, due to good performance and lowest cost.

6 Summary and Outlook

This work investigated the suitability of various distance measures in surrogate modeling for the optimization of several permutation problems. It was shown, that each problem class or instance may require a different distance measure. Correlation between distance and fitness values (LFDC) proved to be a poor way of selecting a distance measure for a given problem class or instance. On the other hand, integrating the selection of a measure into the MLE process of a Kriging model proved to be a very well performing and robust approach.

Further research may focus on learning distance measures for Kriging-based models in combinatorial spaces. Learning of correlation functions with Genetic Programming is not new [12]. Also, distance measures have been evolved with GA [8] in the context of string matching. Combining both ideas to evolve better measures for distance-based models may thus be an interesting path to follow.

If interpretable distance measures evolve, this may also give interesting insight into the underlying problems.

Acknowledgments This work has been kindly supported by the Federal Ministry of Education and Research (BMBF) under the grant CIMO (FKZ 17002X11).

References

1. Abdul-Razaq, T., Potts, C., Wassenhove, L.V.: A survey of algorithms for the single machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics* 26(23), 235–253 (1990)
2. Allahverdi, A., Ng, C., Cheng, T.E., Kovalyov, M.Y.: A survey of scheduling problems with setup times or costs. *European Journal of Operational Research* 187(3), 985–1032 (2008)
3. Altenberg, L.: Fitness distance correlation analysis: An instructive counterexample. In: Bäck, T. (ed.) *ICGA*, pp. 57–64. Morgan Kaufmann (1997)
4. Bartz-Beielstein, T., de Vegt, M., Parsopoulos, K.E., Vrahatis, M.N.: Designing particle swarm optimization with regression trees. Technical Report CI-173/04, Universität Dortmund, (Mai 2004)
5. Beasley, J.E.: OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society* 41(11), 1069–1072 (1990)
6. Burkard, R.E.: Quadratic assignment problems. *European Journal of Operational Research* 15(3), 283–289 (1984)
7. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization* 10(4), 391–403 (1997)
8. Camacho, D., Huerta, R., Elkan, C.: An evolutionary hybrid distance for duplicate string matching. Technical report, Universidad Autonoma de Madrid (2008)
9. Campos, V., Laguna, M., Martí, R.: Context-independent scatter and tabu search for permutation problems. *INFORMS Journal on Computing* 17(1), 111–122 (2005)
10. Caprara, A.: Sorting by reversals is difficult. In: *Proceedings of the First Annual International Conference on Computational Molecular Biology, RECOMB 1997*, pp. 75–83. ACM, New York (1997)
11. Forrester, A., Sobester, A., Keane, A.: *Engineering Design via Surrogate Modelling*. Wiley (2008)
12. Gagné, C., Schoenauer, M., Sebag, M., Tomassini, M.: Genetic programming for kernel-based learning with co-evolving subsets selection. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) *PPSN 2006. LNCS*, vol. 4193, pp. 1008–1017. Springer, Heidelberg (2006)
13. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. *Communications of the ACM* 18(6), 341–343 (1975)
14. Hutter, F.: *Automated configuration of algorithms for solving hard computational problems*. PhD thesis, University of British Columbia (2009)
15. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1(2), 61–70 (2011)
16. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13(4), 455–492 (1998)
17. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: *International Conference on Genetic Algorithms*, pp. 184–192. Morgan Kaufmann Publishers Inc. (1995)

18. Kallel, L., Schoenauer, M.: Fitness distance correlation for variable length representations. Technical report, Ecole Polytechnique (1996)
19. Kendall, M., Gibbons, J.: Rank correlation methods. A Charles Griffin Book. E. Arnold (1990)
20. Lee, C.: Some properties of nonbinary error-correcting codes. *IRE Transactions on Information Theory* 4(2), 77–82 (1958)
21. Li, R., Emmerich, M.T.M., Eggermont, J., Bovenkamp, E.G.P., Back, T., Dijkstra, J., Reiber, J.: Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis. In: *Congress on Evolutionary Computation*, pp. 2764–2771. IEEE (2008)
22. Moraglio, A., Kattan, A.: Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In: Merz, P., Hao, J.-K. (eds.) *EvoCOP 2011*. LNCS, vol. 6622, pp. 142–154. Springer, Heidelberg (2011)
23. Moraglio, A., Kim, Y.-H., Yoon, Y.: Geometric surrogate-based optimisation for permutation-based problems. In: Krasnogor, N., et al. (eds.) *Genetic and Evolutionary Computation Conference*, pp. 133–134. ACM (2011)
24. Reeves, C.R.: A genetic algorithm for flowshop sequencing. *Computers & Operations Research* 22(1), 5–13 (1995)
25. Reeves, C.R.: Landscapes, operators and heuristic search. *Annals of Operations Research* 86, 473–490 (1999)
26. Reinelt, G.: TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing* 3(4), 376–384 (1991)
27. Schiavinotto, T., Stützle, T.: A review of metrics on permutations for search landscape analysis. *Computers & Operations Research* 34(10), 3143–3153 (2007)
28. Serpell, M., Smith, J.E.: Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. *Evolutionary Computation* 18(3), 491–514 (2010)
29. Sevaux, M., Sörensen, K.: Permutation distance measures for memetic algorithms with population management. In: *Metaheuristics International Conference*, pp. 832–838 (2005)
30. Taillard, E.: Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research* 47(1), 65–74 (1990)
31. Voutchkov, I., Keane, A., Bhaskar, A., Olsen, T.M.: Weld sequence optimization: The use of surrogate models for solving sequential combinatorial problems. *Computer Methods in Applied Mechanics and Engineering* 194(30-33), 3535–3551 (2005)
32. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. *Journal of the ACM (JACM)* 21(1), 168–173 (1974)
33. Wikipedia. Longest common substring problem — wikipedia, the free encyclopedia (2014) (Online; accessed March 26, 2014)
34. Zaefferer, M., Stork, J., Friese, M., Fischbach, A., Naujoks, B., Bartz-Beielstein, T.: Efficient global optimization for combinatorial problems. In: *Genetic and Evolutionary Computation Conference* (accepted 2014) (preprint)

Boosting Search for Recursive Functions Using Partial Call-Trees

Brad Alexander and Brad Zacher

School of Computer Science, University of Adelaide, 5005, Australia

bradley.alexander@adelaide.edu.au

brad.zacher@alumni.adelaide.edu.au

<http://www.cs.adelaide.edu.au/~brad>

Abstract. Recursive functions are a compact and expressive way to solve challenging problems in terms of local processing. These properties have made recursive functions a popular target for genetic programming. Unfortunately, the evolution of substantial recursive programs has proven difficult. One cause of this problem is the difficulty in evolving both correct base and recursive cases using just information derived from running test cases. In this work we describe a framework that exploits additional information in the form of partial call-trees. Such trees - a by-product of deriving input-output cases by hand - guides the search process by allowing the separate evolution of the recursive case. We show that the speed of evolution of recursive functions is significantly enhanced by the use of partial call-trees and demonstrate application of the technique in the derivation of functions for a suite of numerical functions.

Keywords: Recursion, Genetic Programming, Call-Tree, Adaptive Grammar.

1 Introduction

Recursion is a compact and expressive way to define solutions to challenging problems in terms of local processing. The brevity and power of recursion have made the evolution of such functions a popular target for genetic programming (GP) [5], [9], [6]. Unfortunately, the evolution of non-trivial recursive functions through GP has proven difficult in practice [1]. One cause of this difficulty is the need to simultaneously evolve correct code for base and recursive cases [1], [7] before a good fitness score is achieved.

Several approaches to improve search been tried. These have included: the use of niches to preserve diversity during search [7]; the automated discovery and separate evolution of base cases [6]. Other work has narrowed the search-space using templates expressing common patterns of recurrence [10,11].

However, while these approaches are beneficial, a central problem remains that the test cases used to evaluate fitness in GP provide poor guidance in the search for the *recursive* clause in recursive functions. In this work we improve search using additional information in form of *partial call-trees*. We show how this

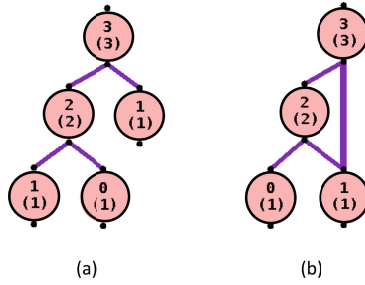


Fig. 1. An example call tree for a Fibonacci function (a) and an equivalent graph (b) with different ordering of children and shared child nodes

extra information can substantially improve GP search for recursive functions by allowing code for recursive calls to be separately evolved.

1.1 Call Trees

A call tree, an example of which is shown in Fig 1(a), is a diagram often used when informally reasoning about recursive problems. Part (b) of Fig 1 shows a graph which is equivalent to part (a) from the point of view of our framework but faster for the user to draw¹. Each node in the call tree contains the parameter(s) of the call and each child represents the the sub-calls made by that node. The return value from a call can also be included in a call tree node. In Fig 1 these return values are shown in brackets. In this work we only require the user to provide call-trees with return values for *some* nodes. Moreover, not every child call of each node is required and the tree can even be disjoint if the user desires. Our framework is designed to require no more information from the user than they might already create in the first, informal, stages of reasoning about a recursive function. This extra information can then be harnessed to boost GP search by allowing for the separate evolution of the code for the recursive case.

1.2 Contributions

We describe a framework that extracts information from a call-tree to boost GP search. We demonstrate that our framework, which we name: Call-Tree-Guided Genetic Programming (CTGGP), significantly improves the speed of search over conventional GP search on a range of benchmarks. Moreover, we show that the structure of partial call trees can be used to guide the choice of grammars which further restricts the search space. This restricted search space permits the evolution of functions with quite complex behaviour including functions that make their calls in loops.

¹ For the sake of brevity we will refer to both as trees in this article since our framework treats both equivalently.

The rest of this article is structured as follows. In the next section we outline related work. In section 3 we describe CTGGP and how it processes its inputs to produce recursive functions. In section 4 we describe our experimental setup including the benchmarks and grammars we use. In section 5 we present our results and, finally, in section 6 we summarise our findings and canvas future work.

2 Related Work

The difficulty in evolving the base and recursive case of recursive functions has been recognised by several authors [1], [7], [6]. Nishiguchi and Fujimoto [7] improved search by allowing less fit individuals to be preserved in a niche to maintain diversity. Moraglio et al. [6] showed that separate evolution of code for the base-case significantly improved search. This work is the most similar to ours in using the idea of separating the search of the cases. However, our work differs by separately evolving the recursive-case rather than base-case and, thus, is complementary to Moraglio's work.

Other authors have improved performance by reducing the search space by: restricting grammars to common patterns of computation [11]; or by adapting grammars to the application [10].

Finally, the use of direct inference about the relationship between recursive calls is a feature of inductive programming [3,4]. This work has been very effective in performing direct search. However, such work is typically restricted to list functions where operators can be more readily inferred from I/O cases.

3 CTGGP

The search algorithms in CTGGP take a partial-call-tree as input and produce a recursive function in C as output. In our current experiments, CTGGP is restricted to discovering functions of one or two integer parameters producing an integer result². The input tree is quickly authored by the user using a Java GUI forming part of the system.

The search process in CTGGP is built on Grammatical Evolution [8] using the C++ GELib (0.26) distribution. Grammatical Evolution (GE) is a GP framework that allows the user to specify an arbitrary grammar for a target language. GE is then able to use this grammar to guide a genotype-to-phenotype mapping that maps a bit-string genome into a syntactically correct individual program. Because this mapping is guided by the grammar, all individuals produced are syntactically correct in the target language.

CTGGP's search process has two phases. These are shown in Fig 2. Phase one evolves code determining the recursive calls of the target function and also selects grammars to be used in phase two. Phase two evolves the remainder of the target function. The second phase is a conventional application of GE using the

² Though we foresee no barriers to generalisation to other types.

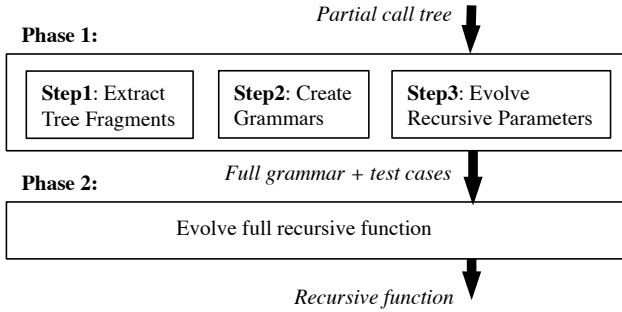


Fig. 2. Evolutionary process of CTGGP

input and output values (the ones in brackets in Fig 1) as tests in the evaluative function. As such we won't describe phase two in further detail. The primary contribution of this article is in phase one and we describe this next.

3.1 Phase One Search

Phase one has three steps. Step one, extracts tree fragments from the partial call-tree provided by the user. Step two, adapts and chooses grammars for phase one *and* phase two search. Step three uses GE to evolve the code that determines the parameters to the recursive calls in the recursive case. The code from step three will be embedded in the candidate programs generated in phase two. We outline each step in phase one next.

Step One: Producing Tree Fragments. Step one traverses the user-supplied call tree and, for each non-leaf node, i produces a target tree fragment of the form (tp_i, tc_i) where tp_i is the input parameter to the parent node and target child list: tc_i is an unordered list of the child node input parameters. To illustrate: the list of target fragments for both parts of Fig 1 is

$$[(3, [1, 2]), (2, [0, 1])] \tag{1}$$

These target fragments are compared to those produced by candidate individuals in step three of phase one.

Step Two: Grammar Production. Step two inspects the call tree provided by the user and, with confirmation from the user, builds the grammars to guide phase one and phase two search. CTGGP guesses the grammars to be used based on the number of children per node in the tree. If there are two children per-node as there are in Fig 1 then it will guess that there are two recursive calls. In contrast, if there are a variable number of children per node it will guess that the recursive calls take place in a loop. CTGGP will also guess the number of

<pre> <expr_root> ::= <var> <op> <digit> <digit> <op> <var> <op> ::= - * + / <digit> ::= 0 1 2 <big_digit> <big_digit> ::= 3 4 5 <bigger_digit> <bigger_digit> ::= 6 7 <huge_digit> <huge_digit> ::= 8 9 <var> ::= x </pre>	<pre> <expr_root> ::= <guard> -- <param> <guard> ::= (<var> <i>) (<var> % i) == 0 (TRUE) <param> ::= i <op> <var> <var> <op> i <var> - i <op> ::= * + / <var> ::= x </pre>
(a)	(b)

Fig. 3. Phase one grammars (a) for fixed numbers of calls and (b) for calls in a loop

base cases based on the number of calls. Since the tree provided by the user is not required to be complete, these initial guesses could be wrong and so the user is asked to confirm and correct the number of call and base cases.

Two sets of grammars are produced. The basic grammar choices for phase one are shown in Fig 3. Part (a) shows the grammar for the parameters when the number of calls is fixed. Note there is only one variable allowed: the input variable x to the recursive function itself. Part (b) shows the grammar used when the calls take place in a loop. This grammar has two parts: a *guard*, which contains a condition determining when a recursive call can be made, and *param* forming the call's parameter.

Figure 4 shows three examples of recursive grammars used in phase two search. The bolded **param** and **guard** keywords indicate where the code from phase one search is inserted. The examples given are the body: of a Fibonacci function (part (a)); of a simple linear-recursive function (part (b)) and of a function making calls in a loop (part (c)). Note, the grammar in part (c) assumes that the loop is bounded by a control variable, c , whose initial value is passed in as a parameter to a helper function: *aux*. The presence of a control variable and guard allows interesting enumerations to be expressed – beyond those found so far in the GP literature. Also note that no explicit production for **op** is given in part (c). In this case the syntax for **op** is a pairing of an operator (e.g. $+$) and its left-identity (e.g. 0). The variable *result* is initialised with this left-identity in the section of code abbreviated as **preamble** in part (c).

Step Three: Evolving the Recursive Parameters. Step three performs GE search to evolve parameter expressions of the recursive calls using the phase one grammar. During this search individuals are evaluated by executing them against inputs tp_i to produce a list of one or more child parameter expressions: cc_i these can then be compared to their corresponding target child list tc_i extracted in step one. A penalty is assessed in proportion to the mismatch between each tc_i and cc_i all of these penalties are summed to derive the total penalty for the individual. Thus, in our Fibonacci example, a candidate individual consisting

<pre> <expr_root> ::= if(<var> < <digit>){ return <lit>; }else{ return <expr1> <op> <expr2>; } <expr1> ::= <rec1> (<rec1> <op> <lit>) (<lit> <op> <rec1>) <expr2> ::= <rec2> (<rec2> <op> <lit>) (<lit> <op> <rec2>) <rec1> ::= recurse(param1) <rec2> ::= recurse(param2) <op> ::= - * + <lit> ::= <digit> <var> ... as per phase 1 grammar... </pre> <p style="text-align: center;">(a)</p>	<pre> <expr_root> ::= if (<var> < <digit>) { return <lit>; } else { return <expr1>; } <expr1> ::= <rec1> <rec1> <op> <lit> <lit> <op> <rec1> <rec1> ::= recurse(param) <op> ::= - * + <lit> ::= <digit> <var> ... as per phase 1 grammar.. </pre> <p style="text-align: center;">(b)</p>	<pre> <expr_root> ::= aux(x,<small_digit>) -- int aux(int x, int c){ .. preamble .. if(<var> <rel> <small_digit>){ return <lit> }else{ .. preamble ... for(i=<rl>; i< <ru>; i++){ if(guard) result = result <op> aux(param,i); } return result } <rl> ::= c <lit> <ru> ::= (<var> + <digit>) (<var> - <digit>) (<digit> - <var>) <var> <rel> ::= < > </pre> <p style="text-align: center;">(c)</p>
--	--	---

Fig. 4. Phase two grammar for the parameter to the recursive call for Fibonacci. The param function where the phase one grammar will be substituted is marked in bold. Note the **digit** and **var** productions are the same as for the phase one grammar.

of just one parameter expression $x-1$ generates for the tp_i in (1) above the candidate child lists: $[[2], [1]]$, with $[2]$ being generated from the input 3 and $[1]$ being generated from 2. After candidates cc_i are generated our phase 1 evaluative function gauges the match between the cc_i and their corresponding targets tc_i . In our example this means we try to match $[1, 2]$ with $[2]$ and $[0, 1]$ with $[1]$. Every match attempt generates a penalty by the **assess_match** procedure shown in Fig 5. This procedure works by repeatedly: finding the numerically closest pair of values between *targetlist* and *candlist* (a process labeled *bestMatch* in Fig 5); calculating a distance penalty; and removing matched items from both lists as it goes. The main loop terminates when one of the lists is depleted. After the loop, the presence of surplus target expressions means that the candidate list didn't cover the target list (i.e. there weren't enough calls) and a large penalty is applied. Conversely, surplus candidate expressions could, benignly, indicate that user didn't supply all of the child nodes when drawing the partial tree. A small penalty is applied in proportion to the number of extra candidates. In our example, when matching candidate $[2]$ with target $[1, 2]$ **assess_match** will match the 2's (penalty: 0) and then have $[1]$ remaining in the target list with a total penalty of *BIG-PENALTY*. The same penalty is assessed for the match between $[1]$ and $[0, 1]$ and penalties are summed resulting in a total penalty of $2 \times \text{BIG-PENALTY}$. The values of *BIG-PENALTY* and *SMALL-PENALTY* are set so that *BIG-PENALTY* is larger than any expected difference in result values and *SMALL-PENALTY* is much smaller than 1.0. The penalties are summed to form an evaluative score for an individual. Evolution proceeds until either an

```

assess_match(targlist,candlist)
  penalty = 0;
  while ( $|targlist| > 0 \wedge |candlist| > 0$ )
    (targlisti, candlistj) = bestMatch(targlist, candlist)
    penalty = penalty +  $|targlist_i - candlist_j|$ 
    targlist = targlist \ targlisti
    candlist = candlist \ candlistj
  end while
  penalty = penalty +  $|targlist| * BIG\_PENALTY$ 
  penalty = penalty +  $|candlist| * SMALL\_PENALTY$ 
  return penalty
end

```

Fig. 5. Procedure to assess match between target and candidate lists

individual with zero penalty is found or a set number of generations has elapsed. Note, that for grammars with more than one recursive call, we run step-three search once for each recursive call, excluding previously found solutions as we go. Also note that for grammars with loops we apply the guard to restrict the calls made but we have to assume large loop bounds. Loop-bounds will not be evolved precisely until phase two. This can lead to surplus candidate calls and the small fitness penalty that this entails.

4 Experimental Setup

In our experiments, we compare the search performance of CTGGP to standard GE. The benchmarks for our experiments are, for an integer parameter (n): *factorial* returns the factorial of n ; *odd-evens* returns 0 if $n \bmod 2 = 0$ and 1 otherwise; *log2* finds $\lfloor \log_2 n \rfloor$; *fib* and *fib3* calculates the Fibonacci and Fibonacci-3 number for n ; *lucas* calculates the n th Lucas number; *factorings* returns the number of unique factorings of n . *sums* returns the number of unique sum-decompositions of n . These latter two benchmarks are not trivially coded by humans and are not found elsewhere in the literature on recursive GP.

The input is a small tree drawn by CTGGP's GUI. Trees for three benchmarks are shown in Fig 6. Note how in part (a) and part (b) we have shared some child nodes between sub-trees and in (a) we have included some disjoint single node trees. Trees for our other benchmarks are of very similar size to these.

In both phases of evolution we used GE running on an underlying steady-state GA with tournament selection. The replacement probability used was 0.25 and probabilities for crossover and mutation were, respectively, 0.9 and 0.01. In both phases individuals were evaluated by using scripts to insert evolved code into test harnesses and running Tiny-C-Compiler [2] (TCC) to quickly generate binaries.

For phase one evolution we used small population of 100 individuals running for ten generations. For phase two we ran with populations of 200 for the

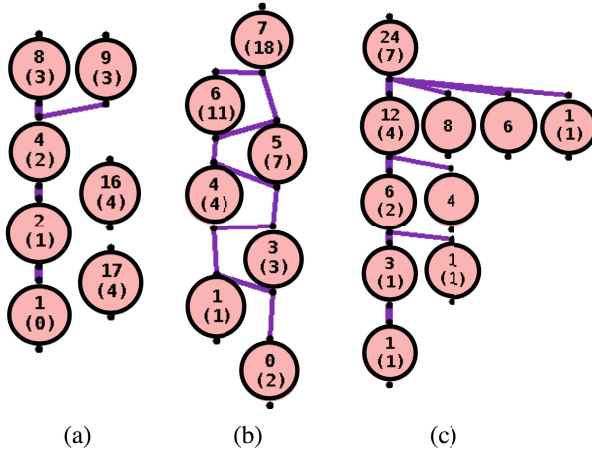


Fig. 6. A selection of input trees: log2a, part (a), lucas part (b), factorings part (c)

smaller *factorial*, *log2* and *oddeven* benchmarks, and with 1000 for the remaining benchmarks. All phase two experiments were run for 100 generations.

As a control, we ran all benchmarks in a single conventional GE phase with with a grammar including the recursive case clauses used in phase one. Note, to better expose the impact of CTGGP these larger grammars were specialised to each benchmark so that the only difference between the conventional GE and the CTGGP runs is the former is required to evolve the recursive parameter code along with the rest of the code³. When evaluating CTGGP we sum the total number of evaluations required for *both* phases. All benchmarks were run on a 2.4GHz Intel core i7 with 4GB of RAM.

5 Results and Discussion

We ran both phases of CTGGP on all benchmarks 100 times. We did likewise for our single phase comparison benchmarks for conventional GE. Phase one of CTGGP succeeded in finding the correct code for parameters in all benchmarks in all runs.

The results comparing the combined cost of phase 1 and phase 2 of CTGGP with conventional GE are shown in Table 1 The columns show, respectively, the mean number of evaluations, the sample standard deviation of the evaluation count, and the percentage of runs yielding a correct result for conventional GE and CTGGC. Note, that because we limit experiments to 100 generations the value of \bar{x} is a lower bound when not all runs are correct. Statistical significance was assessed using a log-rank test to take account of this truncation.

³ This is perhaps generous to conventional GE which would not be able to select the correct grammar in the absence of tree information.

Table 1. Mean number of evaluations and number of correct answers for raw GE and CTGGP. Significantly better ($p \ll 0.01$) means are marked in bold.

problem	Conventional GE			CTGGP		
	\bar{x}	σ	correct	\bar{x}	σ	correct
factorial	1984	1461	99	436	79	100
oddevens	507	343	100	484	160	100
log2	7756	2138	24	3618	3520	81
fib2	32933	10700	53	2156	256	100
fib3	31375	3437	3	7863	4852	100
lucas	32012	10577	40	11713	7736	99
factorings	28266	15695	60	1937	1071	100
summands	38912	7611	26	24926	11642	91

The data from our runs show that, in most benchmarks, CTGGC, significantly out-performs conventional GE both in terms of the reduced number of evaluations required and the number of times a benchmark was correctly evolved. The only benchmark that did not show a significant improvement was *oddeven* which presented an easier target than other benchmarks, partly because it admits a reasonable diversity of solutions. In contrast, the *log2* benchmark exhibited a tendency to prematurely converge toward locally strong solutions. Likewise, *summands* exhibited similar tendencies as well being sensitive to the upper bound of its loop.

Another observation to be made was the high variance in the number of evaluations required. In CTGGP this is caused by the significant number of runs which found solutions in the first one or two generations.

In terms of experimental run times we observed TCC to be fast and we set the timeouts for phase two runs to be small so the average evaluation time for an individual in both phases is 2 milliseconds. Runtimes for phase one evolution varied from less than five seconds to just over a minute. Runtimes of phase two evolution varied from less than 20 seconds (for factorial) to several minutes (for lucas). A final observation to make is that all of the trees used in these experiments were very easy to draw. This ease of use and the short runtimes are positive indicators for GTGGP's future implementation as a practical tool.

6 Conclusions and Future Work

In this article we have shown that incorporating call-tree information into the GP search process can significantly improve performance at only a small cost in terms of human effort. The work here is most applicable where code is required to implement a completely unknown recurrence and the drawing of a partial call-tree is a natural part of the exploratory process. The usefulness of CTGGP is in automating the non-trivial step of deriving code from this partial call-tree.

This work can be enhanced in several ways. We could exploit the relationships implicit in return values to separately derive code combining results of recursive

calls. We could exploit existing libraries of sequences to express more complex recurrences in loops. We could combine Moraglio's technique for discovering base-cases with ours. Finally, we could integrate hand-drawn-graph-recognition software into CTGGP to allow direct derivation of recursive code from paper sketches, completing the chain from pictures to programs.

References

1. Agapitos, A., Lucas, S.: Learning recursive functions with object oriented genetic programming. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 166–177. Springer, Heidelberg (2006)
2. Bellard, F.: Tcc: Tiny c compiler (2003), <http://fabrice.bellard.free.fr/tcc>
3. Hofmann, M., Kitzelmann, E., Schmid, U.: A unifying framework for analysis and evaluation of inductive programming systems. In: Proceedings of the Second Conference on Artificial General Intelligence, pp. 55–60 (2009)
4. Kitzelmann, E., Schmid, U.: Inductive synthesis of functional programs: An explanation based generalization approach. *The Journal of Machine Learning Research* 7, 429–454 (2006)
5. Koza, J.R., Andre, D., Bennett III, F.H., Keane, M.: *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufman (April 1999)
6. Moraglio, A., Otero, F.E.B., Johnson, C.G., Thompson, S., Freitas, A.A.: Evolving recursive programs using non-recursive scaffolding. In: *IEEE Congress on Evolutionary Computation*, pp. 1–8 (2012)
7. Nishiguchi, M., Fujimoto, Y.: Evolution of recursive programs with multi-niche genetic programming (mngp). In: *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 247–252 (1998)
8. O'Neill, M., Ryan, C.: Grammatical evolution. *IEEE Trans. Evolutionary Computation* 5(4), 349–358 (2001)
9. Spector, L., Robinson, A.: Genetic programming and autoconstructive evolution with the push programming language. In: *Genetic Programming and Evolvable Machines*, pp. 7–40 (2002)
10. Wong, M.L., Leung, K.S.: Evolving recursive functions for the even-parity problem using genetic programming. In: *Advances in Genetic Programming*, pp. 221–240. MIT Press (1996)
11. Yu, T., Clark, C.: Recursion, lambda-abstractions and genetic programming. *Cognitive Science Research Papers-University of Birmingham CSRP*, 26–30 (1998)

Compressing Regular Expression Sets for Deep Packet Inspection

Alberto Bartoli, Simone Cumar, Andrea De Lorenzo, and Eric Medvet

Department of Engineering and Architecture, University of Trieste, Italy

Abstract. The ability to generate security-related alerts while analyzing network traffic in real time has become a key mechanism in many networking devices. This functionality relies on the application of large sets of regular expressions describing attack signatures to each individual packet. Implementing an engine of this form capable of operating at line speed is considerably difficult and the corresponding performance problems have been attacked from several points of view. In this work we propose a novel approach complementing earlier proposals: we suggest transforming the starting set of regular expressions to another set of expressions which is much smaller yet classifies network traffic in the same categories as the starting set. Key component of the transformation is an evolutionary search based on Genetic Programming: a large population of expressions represented as abstract syntax trees evolves by means of mutation and crossover, evolution being driven by fitness indexes tailored to the desired classification needs and which minimize the length of each expression. We assessed our proposals on real datasets composed of up to 474 expressions and the outcome has been very good, resulting in compressions in the order of 74%. Our results are highly encouraging and demonstrate the power of evolutionary techniques in an important application domain.

Keywords: Genetic programming, evolutionary optimization, intrusion detection, traffic classification.

1 Introduction

The ability to generate security-related alerts while analyzing network traffic in real time has become a key mechanism in many networking devices, ranging from intrusion detection systems to firewalls and switches. While early systems classified traffic based only on header-level packet information, modern systems are capable of detecting malicious patterns within the actual packet payload. This *deep packet inspection* capability is usually based on pattern descriptions expressed in the form of *regular expressions*, because fixed strings have become inadequate to describe attack signatures.

Implementing a regular expression evaluation engine capable of analyzing network traffic at line speed is considerably difficult, also because there are usually hundreds or thousands of regular expressions to be analyzed and this set needs to

be periodically updated to address novel attacks. For this reason, there has been a considerable amount of recent proposals aimed at handling the corresponding performance problems. Such proposals have addressed different dimensions of the design space: optimization of the evaluation algorithm in representations of regular expressions based on Deterministic Finite Automata (DFA) [1–3]; DFA representations leading to faster hardware implementations and which require less memory [4–6]; optimization of the hardware implementation of DFA [7]; development of engines suitable for parallel hardware implementation [8, 9].

In this work, we address a different dimension of the design space and propose an approach which complements the existing proposals. Rather than optimizing the steps from the set of regular expressions to their run-time evaluation, we explore the possibility of greatly reducing the size of the set itself. To this end, we use an heuristic approach: rather than attempting to construct a new set of expressions formally equivalent to the original one (but simpler to evaluate at run-time) [10], we aim at constructing a set with the same detection behavior on the traffic of interest. As it turns out, this relaxed problem formulation allows a broad range of compressions and simplifications which would not be possible when insisting on having exactly the very same detection behavior on all possible strings.

Key component of our proposal is an evolutionary search phase based on *Genetic Programming* (GP). We create a population of regular expressions composed of the set of expressions to be simplified and further randomly-generated expressions. We then evolve this population by randomly combining expressions with genetic operators (crossover and mutation) for a predefined number of steps. The evolution is driven by a *multi-objective optimization* strategy aimed at minimizing two *fitness* indexes of each expression taken in isolation: classification errors on the traffic of interest and length of the expression. Finally, we construct a set of regular expressions meant to replace the original one by selecting a subset of the final population. We select this subset with a greedy procedure ensuring that the resulting subset tends to have the same detection behavior on the traffic of interest as the original set.

We assess our proposal on several real sets of regular expressions used in the Snort¹ intrusion detection system—one of the standard testbeds in this specific research field, e.g., [7, 10, 3]. We considered sets with a number of regular expressions ranging from 10 to 474 and an aggregate length ranging from 260 to about 59 742. The results are highly promising: we obtain a decrease in the number of regular expressions and a decrease in aggregate length in the order of 74%, on the average.

2 Our Approach

2.1 Problem Statement

We associate each set of regular expressions R with a numerical *cost* $c(R)$, which models the effort required for applying all expressions in R to a given string.

¹ <http://www.snort.org>

This index should quantify the run-time cost of using R and its actual value depends on the specific technology used [7]. In this work we use the sum of the lengths of all expressions in R as a proxy for $c(R)$, i.e., we set $c(R) = \sum_{r \in R} \ell(r)$, where $\ell(r)$ is the length of the regular expression r represented as a string. It will be clear from the following sections that our approach may be applied with widely differing cost definitions: for example, one could consider the number of states of the Nondeterministic Finite Automata (NFA) implementing each expression [10] as well as the presence of specific hard-to-evaluate constructs [11].

We say that a regular expression r *matches* a string s , denoted $r \leftrightarrow s$, if r extracts at least one non-empty substring of s . We say that a set of regular expressions R matches a string s , denoted $R \leftrightarrow s$, if at least one of the regular expressions $r \in R$ matches s . We say that a set of regular expressions R_1 is *equivalent* to another set R_2 if the set of all the strings matched by R_1 is equal to the set of all the strings matched by R_2 . Given a finite set S of sample strings, we say that R_1 is *S-equivalent* to R_2 if the set of all the strings in S matched by R_1 and R_2 is the same, i.e., $\{s \in S : R_1 \leftrightarrow s\} = \{s \in S : R_2 \leftrightarrow s\}$.

Given a starting set of regular expressions R_s , we generate synthetically from this set a *positive* set S^+ of matched strings and a *negative* set S^- of strings which are not matched. We aim at identifying a *different* set of regular expressions R_f such that: (i) R_s and R_f are $(S^+ \cup S^-)$ -equivalent; and, (ii) $c(R_f) < c(R_s)$.

To solve this problem, we proceed as follows.

1. We generate S^+ and S^- from R_s with the same cardinality. We then randomly partition each set in three subsets to be used in the two next phases of the algorithm and for testing: i.e., we partition S^+ in $S^+_{\text{evolution}}$, $S^+_{\text{selection}}$ and S^+_{testing} , and the same for S^- . In this work we chose to use three equally-sized subsets, but different choices are possible.
2. In the *evolution* phase, we evolve the starting set of regular expressions R_s with a stochastic procedure based on GP. The evolution is driven by a multi-objective optimization strategy aimed at minimizing two performance indexes of each expression r taken in isolation: errors of r on $S^+_{\text{evolution}}$, $S^-_{\text{evolution}}$ and length of the expression $\ell(r)$. We execute n independent evolutions, each evolution producing a set of regular expressions R_e^i , with $i = 1, \dots, n$.
3. In the *selection* phase, we construct a candidate target set R_f^i based on the set R_e^i generated in the previous phase, with $i = 1, \dots, n$. The construction of each set R_f^i is made with a set coverage algorithm aimed at selecting a subset of R_e^i matching all examples in $S^+_{\text{selection}}$ and no example in $S^-_{\text{selection}}$. The coverage is driven by a greedy strategy aimed at minimizing the cost of R_f^i . We select as target set R_f the set R_f^i with smallest cost.

We emphasize that the evolution phase optimizes performance indexes of each regular expression taken in isolation, while the selection phase optimizes an index resulting from the coordinated effort of all the regular expressions.

We assessed our procedure on several sets of regular expressions used in Snort. For each set R_s , we assessed the generated target set R_f by comparing its cost $c(R_f)$ to the cost of the original set $c(R_s)$. Furthermore, we verified that R_f matches all strings in S^+_{testing} and does not match any string in S^-_{testing} .

The starting set of expressions R_s is obtained from detection rules generated by administrators, each expression in R_s being associated with exactly one detection rule. Transforming R_s to a different set R_f , much cheaper to evaluate at run-time, implies that when R_f matches a given string there is usually no immediate correspondence with detection rules. This issue is intrinsic to any approach aimed at optimizing R_s as a whole, e.g., [10], as opposed to optimizations where the original regular expressions are left unchanged. We remark, though, that identifying the detection rule in order to generate a meaningful alert description may be done rather simply: once R_f has classified a certain packet as a positive, it suffices to apply R_s on that packet. The key observation is that packet processing has to be performed at line speed, while alert description may proceed at a much slower pace. Indeed, this strategy also allows correcting any false positive misclassifications due to the transformation from R_s to R_f —a packet classified as positive by R_f which is actually not matched by any expression in R_s would not generate any alert.

2.2 Representation

We represent each regular expression as an abstract syntax tree. A regular expression r is produced from a tree by concatenating node labels encountered in a depth-first post order visit of the tree. The label of each leaf node is an element from a predefined *terminal set* whereas the label of each branch node is an element from a predefined *function set*.

The terminal set is composed of constants ($\mathbf{a}, \dots, \mathbf{z}, \mathbf{A}, \dots, \mathbf{Z}, \mathbf{0}, \dots, \mathbf{9}, \backslash\mathbf{x}00, \dots, \backslash\mathbf{x}07, -, ?, (,), \{, \}, ., \mathcal{O}, \#, _ , _ , \dots$) and character classes ($\backslash\mathbf{w}, \backslash\mathbf{W}, \backslash\mathbf{d}, \backslash\mathbf{D}, \backslash\mathbf{s}, \backslash\mathbf{S}, \mathbf{a-z}$ and $\mathbf{A-Z}$). The function set is composed of the following operators: the concatenator \cdot , which concatenates its two children (the dot character \cdot represents a placeholder for the children nodes of the corresponding node); the character class operators $[\cdot]$ and $[\hat{\cdot}]$, the non-capturing group $(?:\cdot)$ operator, the capturing group (\cdot) operator, the disjunction $\cdot|$ operator and the greedy quantifiers $(\cdot^*, \cdot^+, \cdot?, \cdot\{\cdot, \cdot\})$.

2.3 Set Equivalence by Sample Strings

An essential component of our heuristic approach is the choice of the sets S^+ , S^- of sample strings to be used for checking the (relaxed) equivalence of the starting set and final set of regular expressions. These samples may be chosen in several ways, for example by using a synthetic *traffic generator* specialized for evaluating deep packet inspection architectures [12]. Another possibility consists in using samples of real traffic explicitly collected for assessing intrusion detection systems [13, 14]. In this paper, we chose to use a simpler approach in which we generate traffic synthetically based solely on the structure of the regular expressions in the starting set R_s , as described below. Further experimentation with traffic generation strategies like those of the cited works is certainly required in order to better validate our results.

For each regular expression $r \in R_s$, we generate k positive strings s such $r(s) = s$ —where $r(s)$ denotes the leftmost non-empty substring of s extracted by r . Then, we generate $k|R_s|$ random strings such that R does not match any of these negative strings. The outcome of the procedure consists of the sets S^+ , S^- , such that (i) $\forall s \in S^+, R_s \leftrightarrow s$, (ii) $\forall s \in S^-, R_s \not\leftrightarrow s$, and (iii) $|S^+| = |S^-| = k|R_s|$.

We generate each positive string s from a $r \in R_s$ as follows. We traverse the tree representation of r (see previous section) in depth-first: each function node generates a string which depends on the node and its children; each terminal node generates a string which depends on the node only. For example, the terminal node `\d` generates a digit with uniform probability; the disjunction node `·|·` generates the first child or the second child string, with equal probability.

We generate each negative string $s \in S^-$ at random. If $R_s \leftrightarrow s$, we drop s and randomly generate a new one. Negative strings have a maximum length of 120 characters.

2.4 Evolution Phase

In this phase, we evolve the starting set of regular expressions R_s with a procedure based on GP and produce a set of regular expressions R_e^i which will be used in the next phase: the whole procedure described in this section is repeated for $i = 1, \dots, n$ and different random seeds. We use an approach which follows closely a proposal for generating automatically regular expressions for *text extraction* based on labelled examples [15]. We summarize the approach in order to provide sufficient background for this work and outline at the end of this section the changes which we applied to the original approach.

The evolutionary search, described below, is based on the NSGA-II [16] multi-objective optimization algorithm. Each candidate solution r has two *fitness* indexes to be minimized: the length $\ell(r)$ of the regular expression and an index $e(r)$ quantifying the classification errors of r on $S_{\text{evolution}}^+, S_{\text{evolution}}^-$. In detail, the index $e(r)$ is defined as:

$$e(r) = \sum_{s \in S_{\text{evolution}}^+} d(s, r(s)) + \sum_{s \in S_{\text{evolution}}^-} d(\emptyset, r(s)) \tag{1}$$

where $d(s_1, s_2)$ is the Levenshtein distance (edit distance) [17] between strings s_1 and s_2 —note that $d(\emptyset, s) = \ell(s)$. In other words, $e(r)$ is the sum of two components: sum of distances between positive strings and what was actually extracted from the positive string; and, sum of distances between the empty string and what was actually extracted from the negative strings. The rationale is that a perfect r should extract exactly s from each $s \in S_{\text{evolution}}^+$ —since positives s have been generated such that $r(s) = s$, with $r \in R_s$ —and should not extract any string from each $s \in S_{\text{evolution}}^-$. We remark that $e(r)$ quantifies *extraction* errors rather classification errors, that is, the desired behavior is described in terms of (possibly empty) substrings to be extracted from sample strings, rather than in terms of two categories of strings. We chose to not deviate from such

formulation because the cited paper argues that fitness definitions based on mere classification could not be adequate to drive the evolutionary search toward the generation of regular expressions with the desired behavior—different fitness indexes could be explored in future work, though.

Each evolutionary search is made on a population of 500 candidate solutions. The initial population consists of all the regular expressions in the starting set R_s and $500 - |R_s|$ regular expressions generated at random. The population evolves for 500 generations, as follows (recall that we execute n independent searches, each producing a set of 500 candidate solutions). Let P be the current population. We generate an evolved population P' as follows: 20% of the regular expressions are generated at random, 20% of the regular expressions are generated by applying the genetic operator “mutation” to regular expressions of P , and 60% of the regular expressions are generated by applying the genetic operator “crossover” to a pair of individuals of P . We select regular expressions for mutation and crossover with a *tournament* of size 7, i.e., we pick 7 regular expressions at random from P and then select the best regular expressions in this set, according to NSGA-II. Finally, we generate the next population by choosing the regular expressions with highest fitness among those in P and P' . The size of the population is kept constant during the evolution. Upon generation of a new regular expression, we check its syntactic correctness: if the check fails, we discard the regular expression and generate a new one. The outcome set R_e^i is set to the final population.

The approach described in this paper differs from the original proposal in [15] in the following points.

1. The initial population is not generated completely at random: it includes all the expressions in the starting set.
2. The terminal set includes more constants: enlarging the cardinality of the terminal set, as well as of the function set, greatly enlarges the size of the solution space.
3. The function set includes the disjunction operator: it is disadvantageous to use in text extraction because it tends to promote overfitting of the labelled examples. Furthermore, the function set includes the greedy quantifiers (\cdot^* , \cdot^+ , $\cdot^?$, $\cdot\{\cdot, \cdot\}$) and does not include possessive quantifiers (\cdot^{**} , \cdot^{++} , $\cdot^{?+}$, $\cdot\{\cdot, \cdot\}^+$). The former are included because largely used in the starting set R_s , the latter are not included because they are often not supported in deep packet inspection tools.

Inclusion of greedy quantifiers with the standard Java engine for processing regular expressions often results in unacceptably long execution times for this form of evolutionary search [15]. For this reason, we used a different engine, internally built with NFA², where the processing cost depends only on the length of the inputs rather than also on the structure of the expression.

² RE2: <https://code.google.com/p/re2>

2.5 Selection Phase

In this phase we construct a candidate target set R_f^i based on the set R_e^i of regular expressions resulting from the i th evolution ($i = 1, \dots, n$) and then select as target set R_f the set R_f^i with smallest cost.

To construct each R_f^i , we consider $S_{\text{selection}}^+$ as a set to be covered by regular expressions in R_f^i (an element of $S_{\text{selection}}^+$ being covered if it is matched by a regular expression in R_f^i). We then execute a *set coverage* procedure aimed at selecting a subset of R_e^i matching all examples in $S_{\text{selection}}^+$ and no example in $S_{\text{selection}}^-$, as follows.

We define the *score* $\mathcal{S}(r, S', S'')$ of a regular expression r on the sets S', S'' as the number of examples in S', S'' which r handles correctly:

$$\mathcal{S}(r, S', S'') = |\{s \in S' : r \leftarrow s\}| + |\{s \in S'' : r \not\leftarrow s\}| \quad (2)$$

Similarly, we define the score $\mathcal{S}(R, S', S'')$ of a set R on the sets S', S'' of regular expressions the number of examples in S', S'' which R as a whole handles correctly:

$$\mathcal{S}(R, S', S'') = |\{s \in S' : R \leftarrow s\}| + |\{s \in S'' : R \not\leftarrow s\}| \quad (3)$$

The greedy set coverage algorithm starts with $R_f^i := \emptyset$, $S' := S_{\text{selection}}^+$ and consists of the following steps:

1. select $r \in R_e^i \setminus R_f^i$ with highest score $\mathcal{S}(r, S', S_{\text{selection}}^-)$;
2. if $\mathcal{S}(R_f^i \cup \{r\}, S_{\text{selection}}^+, S_{\text{selection}}^-) \leq \mathcal{S}(R_f^i, S_{\text{selection}}^+, S_{\text{selection}}^-)$ then terminate;
3. $R_f^i := R_f^i \cup \{r\}$;
4. $S' := S' \setminus \{s \in S_{\text{selection}}^+ : R_f^i \leftarrow s\}$;
5. if $S' = \emptyset$ or $R_f^i = R_e^i$ then terminate, otherwise go to step 1.

In other words, candidates for inclusion in R_f^i are taken from R_e^i and the choice is driven by the score of candidates on $S', S_{\text{selection}}^-$. The strategy is greedy in the sense that once a candidate is chosen it cannot be removed by a later choice.

These steps are followed by further *completion* steps, to be executed in case of termination with $S' \neq \emptyset$. The completion steps consist in a further execution of the above algorithm, this time starting from the R_f^i obtained at the end of the former algorithm (rather than from $R_f^i := \emptyset$) and by selecting candidates from the original expressions R_s —i.e., in step 1, r is chosen in $R_s \setminus R_f^i$ rather than in $R_e^i \setminus R_f^i$. The rationale is that if elements from R_e^i fail to detect some positives, then the missing positives can be detected by some of the original expressions in R_s .

3 Experimental Evaluation

3.1 Datasets

We used several real sets of regular expressions used in the Snort intrusion detection system, which have been collected by the Netbench project [18]. Table 1

Table 1. Datasets

R_s	$ R_s $	$c(R_s)$	k	$ S^+ \cup S^- $
chat.rules.pcre	14	307 105		2940
pop3.rules.pcre	16	265 105		3360
policy.rules.pcre	10	260 105		2100
web-php.rules.pcre	16	400 105		3360
ftp.rules.pcre	35	645 60		4200
spyware-put.rules.pcre	460	16 277 60		55 200
web-activex.rules.pcre	474	59 742 60		56 880

lists these sets, along with their cardinality and their cost (i.e., aggregate length of all the regular expressions in the set). The table also shows the value k we used in the procedure for generating S^+ , S^- for each set R_s and the resulting number $|S^+ \cup S^-| = 2k|R_s|$ of sample strings.

3.2 Results and Discussion

We applied our approach to each dataset R_s and assessed, in each case, the quality of the resulting set R_f with the following indexes. We quantified the cost reduction by computing the *compression* ratio defined as $1 - \frac{c(R_f)}{c(R_s)}$. Concerning the detection behavior, we computed False Positive Rate (FPR, i.e., percentage of strings in S^-_{testing} which are matched by R_f) and False Negative Rate (FNR, i.e., percentage of strings in S^+_{testing} which are not matched by R_f). We also computed *accuracy* as $1 - \frac{1}{2}(\text{FPR} + \text{FNR})$. Of course, R_s exhibits $\text{FPR} = \text{FNR} = 0$ by construction of sets S^+ , S^- . Thus, R_f should also exhibit $\text{FPR} = \text{FNR} = 0$ but coupled with a compression rate close to 100%.

Table 2 shows the results of our experimental evaluation. The table also shows the performance indexes without the completion steps in the selection phase, in order to highlight to which extent these steps improve results.

Table 2. Results

R_s	Without completion steps				With completion steps			
	FPR	FNR	Acc.	$1 - \frac{c(R_f)}{c(R_s)}$	FPR	FNR	Acc.	$1 - \frac{c(R_f)}{c(R_s)}$
chat.rules.pcre	0.0	50.0	75.0	96.10	0.0	0.0	100.0	70.66
pop3.rules.pcre	2.7	0.0	98.7	91.33	2.7	0.0	98.7	91.33
policy.rules.pcre	88.5	0.0	55.9	8.62	88.5	0.0	55.9	8.62
web-php.rules.pcre	24.5	6.3	84.6	67.00	24.5	0.0	87.8	66.50
ftp.rules.pcre	15.9	7.4	88.4	53.96	15.9	0.0	92.2	48.99
spyware-put.rules.pcre	3.3	9.5	93.6	99.01	1.6	0.0	98.3	91.26
web-activex.rules.pcre	0.0	0.0	100.0	99.97	0.0	0.0	100.0	99.97

It can be seen that the average compression ratio amongst the datasets is 74%, but the key result is that the two largest datasets (spyware-put.rules.pcre and web-activex.rules.pcre) can be compressed to less than 1% of the original size—without affecting accuracy significantly.

We also remark that FNR is zero for all the datasets (thanks to the completion steps) and that FPR is very low for 4 on 7 datasets, but can be reduced to zero on all the datasets as discussed in Section 2.1 (it suffices to apply the original R_s only on those strings which are matched by R_f , which still allows exploiting the advantages of compressions because only R_f has to be applied at line speed).

We performed our experiments on an Intel i5-3470 3.20GHz machine with 8 GB RAM: the time required to process a single dataset was of 4 h on the average.

4 Concluding Remarks

Applying large sets of regular expressions to network traffic while operating at line speed is a challenging problem which has been attacked from several perspectives. In this work, we propose a novel approach complementing earlier proposals and assessed its feasibility. We considered the possibility of transforming the starting set of regular expressions to another set of expressions which is much smaller yet classifies network traffic in the same categories as the starting set. Key component of the transformation is an evolutionary search based on GP: a large population of regular expressions represented as abstract syntax trees evolves by means of mutation and crossover, evolution being driven by fitness indexes tailored to the desired classification needs and which minimize the length of each expression. The desired set of expressions is then built with a greedy algorithm which selects from the available expressions a small set matching all positive samples and not matching any negative. We remark that the evolutionary search optimizes each expression taken in isolation, while the selection phase optimizes the performance of the target population.

We experimented with real datasets and the outcome has been very good, resulting in compressions in the order of 74% across all datasets but well above 90% on the bigger datasets composed of hundreds of expressions. Such compressions could be even improved further by applying other proposals to the final result, e.g., by minimizing the number of states of the NFA representing the final set of expressions [10]. While our proposal certainly needs further investigation, in particular, concerning its performance on real network traffic (see Section 2.3), we do believe that our results are highly encouraging and demonstrate the power of evolutionary techniques in an important application domain.

References

1. Yu, F., Chen, Z., Diao, Y., Lakshman, T., Katz, R.H.: Fast and memory-efficient regular expression matching for deep packet inspection. In: Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems, pp. 93–102. ACM (2006)
2. Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., Turner, J.: Algorithms to accelerate multiple regular expressions matching for deep packet inspection. ACM SIGCOMM Computer Communication Review 36(4), 339–350 (2006)

3. Becchi, M., Crowley, P.: An improved algorithm to accelerate regular expression evaluation. In: Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems, pp. 145–154. ACM (2007)
4. Brodie, B.C., Taylor, D.E., Cytron, R.K.: A scalable architecture for high-throughput regular-expression pattern matching. In: ACM SIGARCH Computer Architecture News, vol. 34, pp. 191–202. IEEE Computer Society (2006)
5. Kong, S., Smith, R., Estan, C.: Efficient signature matching with multiple alphabet compression tables. In: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks, vol. 1. ACM (2008)
6. Becchi, M., Cadambi, S.: Memory-efficient regular expression search using state merging. In: INFOCOM 2007, 26th IEEE International Conference on Computer Communications, pp. 1064–1072. IEEE (2007)
7. Meiners, C., Patel, J., Norige, E., Liu, A., Torng, E.: Fast regular expression matching using small TCAM. *IEEE/ACM Transactions on Networking* 22(1), 94–109 (2014)
8. Becchi, M., Crowley, P.: A hybrid finite automaton for practical deep packet inspection. In: Proceedings of the 2007 ACM CoNEXT Conference, p. 1. ACM (2007)
9. Becchi, M., Crowley, P.: Efficient regular expression evaluation: theory to practice. In: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 50–59. ACM (2008)
10. Kosar, V., Korenek, J.: Reduction of fpga resources for regular expression matching by relation similarity. In: 2011 IEEE 14th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 401–402. IEEE (2011)
11. Bispo, J., Sourdis, I., Cardoso, J.M.P., Vassiliadis, S.: Synthesis of regular expressions targeting fPGAs: Current status and open issues. In: Diniz, P.C., Marques, E., Bertels, K., Fernandes, M.M., Cardoso, J.M.P. (eds.) ARCS 2007. LNCS, vol. 4419, pp. 179–190. Springer, Heidelberg (2007)
12. Becchi, M., Franklin, M., Crowley, P.: A workload for evaluating deep packet inspection architectures. In: IEEE International Symposium on Workload Characterization, IISWC 2008, pp. 79–89. IEEE (2008)
13. Shiravi, A., Shiravi, H., Tavallae, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security* 31(3), 357–374 (2012)
14. Black hat USA 2010: SprayPAL: how capturing and replaying attack traffic can save your IDS 1/2 (September 2010)
15. Bartoli, A., Davanzo, G., De Lorenzo, A., Medvet, E., Sorio, E.: Automatic synthesis of regular expressions from examples. *IEEE Computer* (2013) (Early Access Online)
16. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
17. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 10, 707 (1966)
18. Pus, V., Tobola, J., Kosar, V., Kastil, J., Korenek, J.: Netbench: Framework for evaluation of packet processing algorithms. In: Proceedings of the 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems, pp. 95–96. IEEE Computer Society (2011)

Inferring and Exploiting Problem Structure with Schema Grammar

Chris R. Cox and Richard A. Watson

Department of Electronics and Computer Science
University of Southampton, UK
{c.cox,r.a.watson}@soton.ac.uk

Abstract. In this work we introduce a model-building algorithm that is able to infer problem structure using generative grammar induction. We define a class of grammar that can represent the structure of a problem space as a hierarchy of multivariate patterns (schemata), and a compression algorithm that can infer an instance of the grammar from a collection of sample individuals. Unlike conventional sequential grammars the rules of the grammar define unordered set-membership productions and are therefore insensitive to gene ordering or physical linkage. We show that when grammars are inferred from populations of fit individuals on shuffled nearest-neighbour NK-landscape problems, there is a correlation between the compressibility of a population and the degree of inherent problem structure. We also demonstrate how the information captured by the grammatical model from a population can aid evolutionary search. By using the lexicon of schemata inferred into a grammar to facilitate variation, we show that a population is able to incrementally learn and then exploit its own structure to find fitter regions of the search space, and ultimately locate the global optimum.

Keywords: Generative grammar, compression, evolutionary algorithm, estimation of distribution algorithm, NK fitness landscape.

1 Introduction

The field of natural computing has proposed a variety of both implicit and explicit model-building algorithms that attempt to infer the structure of a problem from populations of above-average-fitness individuals. The intuition is that by using these models evolutionary search can be directed to promising areas of a high-dimensional fitness landscape by recombining multivariate features that are commonly found in fit individuals. The best known implicit model is variation by crossover in sexual genetic algorithms. The seminal work of Holland on “Schema Theorem” [1] and subsequently Goldberg on the “Building Block Hypothesis” [2] showed that, at least in theory, crossover offers a scheme under which fit multivariate patterns known as schemata can grow in frequency in an evolutionary population. Moreover, these schemata can act as genetic building blocks that can themselves be recombined, allowing evolutionary search to be performed

at higher levels of organisation. More recently Estimation of Distribution Algorithms (EDAs) have been proposed that attempt to build explicit, probabilistic models of fit individuals that can be sampled in order to evolve a population through either constructive or perturbative variation (see [3] for a review). These methods have been shown to be competent at solving many different types of problem, including problems that have proven difficult for crossover-based GA's.

In this paper we present an alternative method of explicitly modelling problem structure. Unlike other explicit modelling techniques that build probabilistic models of a sample population using statistical inference, we build a *lossless* model using grammatical inference. By a lossless model we mean a model from which the original samples can be recreated without any loss of information. Indeed, we will show that the grammar induction algorithm used here is a type of lossless compression algorithm that identifies a hierarchy of genetic schemata with which the sample population can be more compactly described. A key feature of the “schema grammar” we introduce is that it generates combinatorial as opposed to sequential languages, with the rules it encodes producing unordered sets of symbols rather than ordered sequences or strings. This sets it apart from traditional sequential grammars and enables grammatical inference in combinatorial problem space for the first time.

We have recently demonstrated the value of this modelling technique in solving synthetic building block problems, showing that the simple modular structure in these problems can be correctly inferred and then reused to facilitate superior time complexity in global search [4]. In the present paper we investigate NK-landscape problems [5], which are irregular and unpredictable but contain an inherent statistical structure. We show that by information about this structure can be learned from a sample population of fit individuals using schema grammar, and that by using the the lexicon of schemata inferred into a grammar to facilitate variation, a population is able to locate fitter regions of the search space and ultimately locate the global optimum.

2 Compression Evolutionary Algorithms

Our objectives when building a model of fit individuals in a population are to identify any inherent structure within the fitness landscape, visible as variable dependencies, and exploit it within the evolutionary search process. Toussaint suggested that one way of achieving this is using Compression Evolutionary Algorithms [6]. By compressing a sample population of phenotypes any dependencies that are present in the population are factored into the structure of the compression model, and what is left is a compressed, decorrelated representation. The idea is that if we consider this compressed representation to be a genotype, and the compression model a genotype-phenotype map, then random perturbation of the genotype will produce phenotypes that obey the dependencies of the problem space. The concept is somewhat similar to the Grammatical Evolution (GE) techniques used in genetic programming [7] in which genetic programs are evolved using a grammar as a genotype-phenotype map, although in the GE

case the grammars are predefined using knowledge of the problem rather than inferred.

Toussaint was able to demonstrate the concept of Compression EA's on variable length problems using sequential grammar inference as the compression model, however the constraints of sequential grammars limit the practical usefulness of the technique. Specifically, only sequentially-contiguous variable dependencies can be modelled, and it is not possible to infer the length constraints or positional structure of a problem.¹ The approach we outline here is an type of Compression Evolutionary Algorithm, also using grammar inference as a compression model, however we overcome the limitations described above by using a set grammar that can produce combinatorial rather than sequential languages, and a genetic encoding with no intrinsic sequential order. This allows us to apply Compression Evolutionary Algorithms to a wider range of evolutionary problems.

3 Schema Grammar

Schema grammar is able to represent and compress any combinatorial expressions that can be encoded as an unordered set of *terminal symbols* which represent the “alphabet” of the problem space. In this paper we will consider n -dimensional binary spaces only. In order to encode genotypes and schemata in the required way we adopt the *Messy GA* encoding of Goldberg et al [9]. The encoding offers an alternative scheme to bit strings for addressing n -dimensional binary spaces using order-independent sets of $\langle locus|allele \rangle$ tuples, for example the bit string 0110 can be represented by the set $\{\langle 0|0 \rangle, \langle 1|1 \rangle, \langle 2|1 \rangle, \langle 3|0 \rangle\}$. In an n -dimensional problem the complete alphabet of terminal symbols, Σ , is just the set of all possible alleles:

$$\Sigma = \{\langle \lambda|\alpha \rangle : \lambda \in \{0, \dots, n\}, \alpha \in \{0, 1\}\} \quad (1)$$

The grammar is a type of context-free grammar (CFG) similar to the context-free grammar codes that are typically used for sequential compression (see [10] for a review). It has *straight-line* properties such that each *non-terminal symbol* (variable) in the grammar is only associated with one production rule and there are no loops in production. These properties ensure that productions are deterministic: in our context this ensures a surjective mapping from genotype to phenotype. Production rules in the grammar are expressed in terms of set membership relations, so using the previous example we can define a non-terminal symbol that represents an individual genotype g using the production rule:

$$g \rightarrow \{\langle 0|0 \rangle, \langle 1|1 \rangle, \langle 2|1 \rangle, \langle 3|0 \rangle\}$$

The encoding also provides a very natural way of representing variable interactions as genetic schemata, including those that may overlap in locus or allele

¹ The meta-Grammar Genetic Algorithm described in [8] manages to apply sequential grammars to solving concatenated trap problems by using “wrapping operators” and pre-defined GE grammars to overcome these limitations.

Table 1. An example grammar extract showing two genotypes that are specified using schemata. The phenotype expansion of each rule is shown in the second column using standard schema notation. Note that the example shown does not constitute a compact representation of the two genotypes - it is used to illustrate the nature of the G-P mapping.

$g_0 \rightarrow \{s_0, \langle 2 0 \rangle, \langle 3 0 \rangle\}$	$g_0 \xRightarrow{*} 110011$
$g_1 \rightarrow \{s_3, \langle 0 0 \rangle, \langle 2 1 \rangle, \langle 3 1 \rangle, \langle 4 0 \rangle\}$	$g_1 \xRightarrow{*} 011100$
$s_0 \rightarrow \{s_1, s_2\}$	$s_0 \xRightarrow{*} 11^{**}11$
$s_1 \rightarrow \{\langle 0 1 \rangle, \langle 5 1 \rangle\}$	$s_1 \xRightarrow{*} 1^{****}1$
$s_2 \rightarrow \{\langle 1 1 \rangle, \langle 4 1 \rangle\}$	$s_2 \xRightarrow{*} *1^{**}1^*$
$s_3 \rightarrow \{\langle 1 1 \rangle, \langle 5 0 \rangle\}$	$s_3 \xRightarrow{*} *1^{***}0$

space. Moreover, the recursive properties of production rules allow schemata to be modelled using multiple levels of sub-schemata, which can be a far more compact representation if those sub-schemata are used in multiple places. These qualities are illustrated in Table 1 below.

We use each instance of schema grammar to represent a genetic population with the symbol table of the grammar providing a surjective mapping of genotypes G to phenotypes P . We specify the mapping as the recursive expansion of the production rules associated with each genotype (denoted using $\xRightarrow{*}$):

$$P = \{p : g \xRightarrow{*} p, g \in G\} \tag{2}$$

Similarly, we say that the phenotype mapping of each schema symbol in S is simply the recursive expansion of the production rules associated with each symbol. It is through this mapping mechanism that we later create genetic variation operators for evolutionary search. The phenotype mappings for both complete genotypes and individual schemata is shown in the second column of Table 1.

We can now formally define schema grammar as the 5-tuple:

$$\mathbb{G}_{Schema} = (V, G, S, \Sigma, R), \text{ where:} \tag{3}$$

- V is a set of non-terminal symbols (variables), each of which defines a sub-language of \mathbb{G}_{Schema} , where $V \ni \{G, S\}$ and $G \cap S \in \emptyset$
- G is a non-empty set of non-terminal symbols representing the genotypes in the sample population
- S is a set of non-terminal symbols representing genetic schemata, which may be empty
- Σ is the set of terminal symbols specified in equation (1)
- R is a finite set of production rules that relate each non-terminal symbol to the expansion of an unordered set of symbols, such that $R : V \rightarrow \{V \cup \Sigma\}^*$

4 Grammar Inference

In this section we describe an offline lossless compression algorithm that is able to infer an instance of schema grammar from a population of sample genotypes. The algorithm we use is based on an adaptation of two existing sequential compression algorithms that use grammar codes: *RE-PAIR* (recursive pairing), an offline algorithm from Larsson and Moffat [11] and *SEQUITUR*, an online algorithm from Nevill-Manning and Witten [12]. The central principle of both our algorithm and the sequential algorithms from which it derives is to infer dependency from frequency of co-occurrence. In sequential compression we define co-occurrence as two symbols appearing next to each other (in order) in a string. In schema grammar we define co-occurrence as two symbols being members of the same set, which is order independent. For example, in the sequential production rule $x \rightarrow abc$ the two co-occurrences present are ab and bc . In schema grammar the co-occurrences in the production rule $x \rightarrow \{a, b, c\}$ are the 2-combinations $\{a, b\}, \{a, c\}, \{b, c\}$.

Compression starts by creating a grammar with no schemata ($S = \emptyset$) and a direct mapping of genotypes to phenotypes using the Messy-GA encoding. The process proceeds iteratively in a way analogous to *RE-PAIR*: on each iteration the two most frequently co-occurring symbols in the genotype encodings G are identified, choosing randomly if two co-occurrences have the same frequency. A new non-terminal symbol is created that expands to the symbol pair and is added to S . All co-occurrences of the two symbols in the samples are then substituted with a single occurrence of the new non-terminal symbol. This process continues recursively until no two symbols co-occur anywhere in the grammar more than once. Larger schemata are formed by the recursive substitution of non-terminal symbols.

We also adopt the rule utility constraints of *SEQUITUR* which allow unnecessary nested hierarchies of symbol pairs to be collapsed into larger symbols. The procedure is simply implemented by taking any non-terminal symbols that

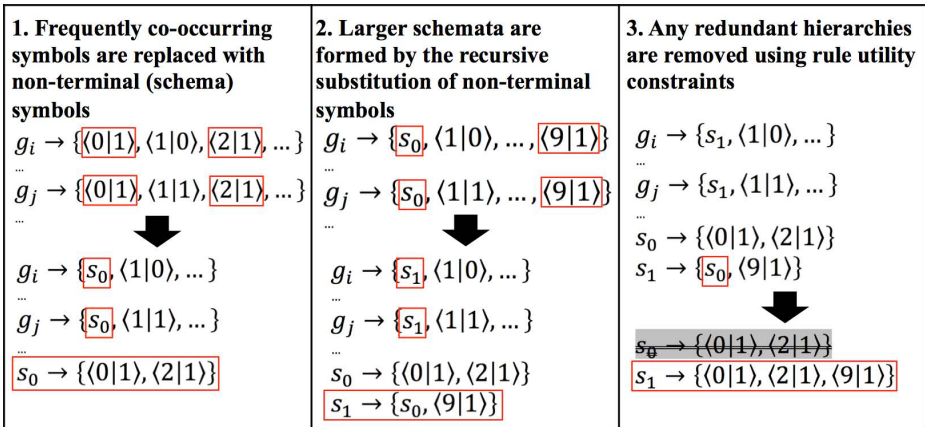


Fig. 1. Illustration of the offline compression algorithm

are referenced only once in the production rules and reversing the symbol substitution. This situation can occur when a particular combinatorial pattern of size $|s| > 2$ only ever appears whole in the samples, with subsets of the pattern not appearing more frequently. Rather than creating a nested hierarchy of $|s| - 1$ non-terminal symbols only one is required. The end result of the process is a compressed, decorrelated genotype representation and a G-P map implemented using a hierarchy of schemata. The operation of the algorithm is illustrated in Fig 1.

5 Inferring and Exploiting Problem Structure

In this section we investigate the ability of the grammar to infer problem structure from fit individuals on NK-landscapes [5], and then exploit the structure to improve evolutionary search. NK-landscape problems use a simple bit-string representation for candidate solutions and are parameterised by n , which specifies the problem size (in bits), and k , which specifies the number of “neighbours” of each bit. We use the nearest neighbour variant of the problem such that the neighbours of each bit overlap in an ordered way (see [5] for details), but shuffle the bits to remove any explicit sequential linkage. By varying neighbourhood size k we are able to tune the degree of correlation structure present in the problem from a completely correlated, unimodal landscape (where $k = 0$) to a completely uncorrelated, random landscape ($k = n - 1$).

Fitness is calculated as the sum of individual bit fitnesses, each of which is a function of its own state and that of the neighbours to which it is connected. A lookup table f_i is created for each bit that maps each of the 2^{k+1} possible input states associated with bit i and its neighbours to a random fitness value in the range 0 to 1. We define the fitness F of a n -bit bitstring X with state $(x_0, x_1, \dots, x_{n-1})$ as:

$$F(X) = \frac{1}{n} \sum_{i=0}^{n-1} f_i(x_i, \Omega(i)) \quad (4)$$

Where $\Omega(i)$ is the state of the k neighbours of bit i . Within the framework of schema grammar a phenotype bitstring X is generated from a schema grammar genotype g by the phenotypic expansion of the production rules associated with g (see Section 3), and rendering the resulting set of terminal symbols as a bit string.

In order to identify individuals of above-average fitness we used the *schema search* process described in Algorithm 1 below. Schema search is a stochastic local search process that is able to use the symbols of schema grammar as variation operators. The search operates in phenotype space and implements phenotypic variation using the recursive expansions of each variation symbol (overwriting any symbols occupying the same loci). The set of symbols used for *schema search* specify the search neighbourhood, and when it is used with terminal symbols (i.e. individual allele values) it is equivalent to a stochastic bit-flip hill climber.

Algorithm 1. Schema Search Pseudocode

```

input : initial phenotype bitstring  $p_i$ 
input : variation symbol set  $\Lambda$ 
output: phenotype bitstring  $p_o$ 
 $p_o \leftarrow p_i$ 
for  $\lambda \in \text{RandomPermutation}(\Lambda)$  do
  |  $\text{candidate}_p \leftarrow p_o$ 
  |  $\text{ExpandInto}(\text{candidate}_p, \lambda)$ 
  | if  $F(\text{candidate}_p) > F(p_o)$  then
  | |  $p_o \leftarrow \text{candidate}_p$ 
  | end
end

```

We generated populations of fit individuals on NK-landscapes with varying epistasis k . For each population we randomised phenotypes and use schema search with terminal symbol (bit-flip) variation operators, and we also ensured that each individual was unique in the population. The resulting phenotypes were compressed into an instance of schema grammar using the method detailed in Section 4. We then measured the compressibility of each population using the entropy of the resulting grammar code (as specified in [10]). This is illustrated in Figure 2, together with a control which is the entropy of a grammar induced from random bit strings of the same size. The figure shows that when there is minimal epistasis ($k = 1$), with a significant degree of correlation structure present in the landscape, the sample population is highly compressible with low entropy. As epistasis is increased then the compressibility of the population is reduced until, in the limit, it is no more compressible than a population of random bit strings.

These results suggest that grammar induction is detecting structure where it is present and using it to compress a population. However by themselves the results do not confirm whether it is detecting the “right” structure, or structure that can be exploited to improve evolutionary search. Some further insight can be gained by looking at the schemata inferred from the population. Figure 3 shows the inferred schema hierarchy for a single above-average fitness individual on an NK-landscape (shown unshuffled for presentation). Although the structure is broadly irregular, as may be expected given the randomised nature of the landscape, patterns reflecting the intrinsic neighbourhood structure of the problem can clearly be seen in the grammar, particularly at higher levels. This decomposition suggests that the schema structure in the grammar may be encoding useful neighbourhood structure from the problem landscape. The figure also illustrates the hierarchical nature of the grammar, with larger building blocks forming from multiple building blocks at lower levels.

We then used a multi-scale search algorithm to investigate whether the information contained in the compression model of a fit population could be exploited to aid evolutionary search. We created a population of fit individuals, each initialised using schema search (Algorithm 1) with bit flip variation operators, which were then modelled using schema grammar induction as described

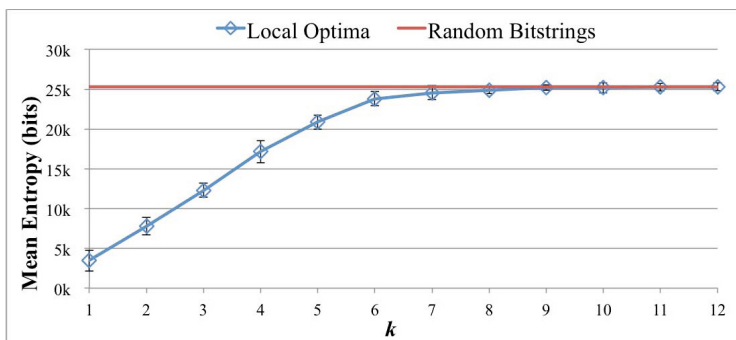


Fig. 2. The average entropy of compressed grammars for populations of local optima with increasing k ($n = 50$, population size = 200). As k increases the compressibility of a population reduces, indicating a reduction in detectable structure.

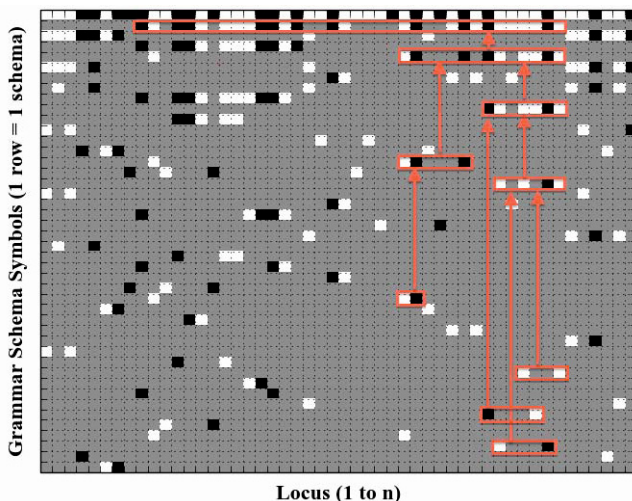


Fig. 3. An example schema grammar hierarchy for a single fit individual on an NK landscape ($n = 50, k = 5$, shown unshuffled). The top row shows the complete individual (black = 0, white = 1 at each locus), with the schemata it is compressed with shown in the rows below (in dependency order). The annotations illustrate part of the dependency hierarchy.

in previous sections. We examined the mutant spectra of the population using the schema symbols in the compressed representation of the population as variation operators (all symbols present in the production rules of each genotype). Although each mutation is a point mutation in compressed genotype space, the hierarchical grammatical expansions of the symbols in phenotype space include mutations with many interacting variables. We compared against two controls: random macro-mutation variation using an instance of the grammar but with shuffled terminal symbols, and uniform crossover and mutation (mutation rate = $1/n$). The results in Figure 4 from a representative problem instance show

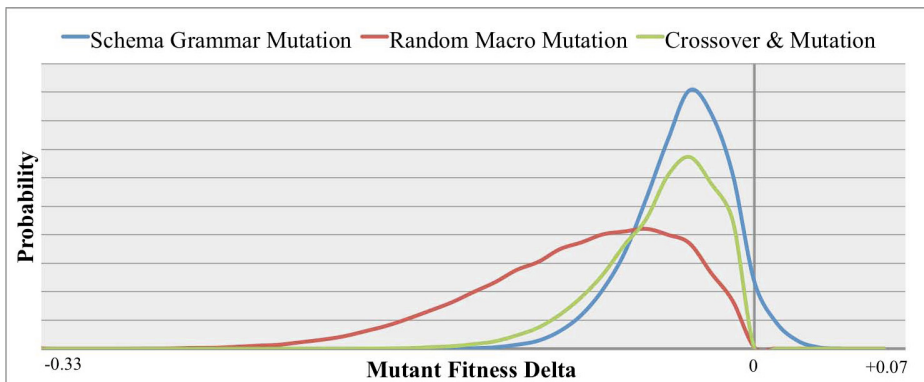


Fig. 4. Mutant spectra of a fit population using schema grammar to facilitate variation, as well as two controls ($n = 50$, $k = 4$, population size = 200). The area under the positive tail of each distribution represents fitness-improving mutations, which were 7.64% of schema grammar mutations, compared to 0.11% of random macro-variations and 0.04% of crossover/mutation variations. The average hamming distance of fitness-improving grammar mutations was 5.66 bits.

that inferred schema structure provides a variation neighbourhood that includes many more fitness-improving mutations than either of the two controls.

We conducted a second experiment to investigate the extent to which the fitness of a population could be improved, solely using the specific schema structure inferred by the grammar to facilitate macro-variation. On each iteration of the algorithm the population was modelled using schema grammar, then schema search was run on the population using the grammar's schema symbols as variation operators. If any beneficial moves were found then this process was repeated, continuing until more moves could be made or the global optimum was located. To help maintain diversity any duplicate phenotypes in the population were re-initialised prior to compression. We ran the algorithm on multiple, random NK-landscapes, with the global optima identified in advance using Pelikan's branch and bound solver [13]. We investigated problems between lengths $n = 20$ and $n = 50$ with k varying in the range 1 – 5, and tested 400 random problem instances for each configuration. In every run of the algorithm the global optimum was successfully located. In less than 2% of problem instances it was found using single-bit hill-climbers (particularly when $k = 1$), however in all other cases the schema structure inferred from the population contained information that allowed search to continue and find fitter regions of the search space, until ultimately the global optimum was found. Further work is planned to test the scalability of these techniques, including comparisons with other model-building solvers such as BOA and LTGA, which are able to efficiently solve nearest-neighbour NK landscapes [14,15].

6 Conclusions

In this paper we have introduced a new class of generative grammar that is capable of modelling combinatorial structure. We have demonstrated that on NK-landscape problems, schema grammar is able to compress a population of individuals using a hierarchy of schema symbols that reflect the intrinsic structure of the landscape. We have also shown that the schemata inferred into the grammar can be exploited by facilitating multi-scale variation during evolutionary search. Our results suggest that the schema grammar is an effective type of compression EA that can demonstrably discover and exploit complex structural regularity in evolutionary problem solving.

References

1. Holland, J.H.: *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. U. Michigan Press (1975)
2. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional (1989)
3. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation* 1(3), 111–128 (2011)
4. Cox, C.R., Watson, R.A.: Solving Building Block Problems using Generative Grammar. In: *Proceeding of the 2014 Conference on Genetic and Evolutionary Computation*, pp. 341–348. ACM (2014)
5. Kauffman, S.A.: *The Origins of Order. Self-organization and Selection in Evolution*. Oxford University Press (1993)
6. Toussaint, M.: Compact representations as a search strategy: Compression EDAs. *Theoretical Computer Science* 361(1), 57–71 (2006)
7. Ryan, C., Collins, J.J., Neill, M.O.: Grammatical evolution: Evolving programs for an arbitrary language. *Genetic Programming*, 83–96 (1998)
8. O'Neill, M., Brabazon, A.: mGGA: The meta-grammar genetic algorithm. *Genetic Programming*, 311–320 (2005)
9. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems* 3(5), 493–530 (1989)
10. Kieffer, J.C., Yang, E.: Grammar-based codes: a new class of universal lossless source codes. *IEEE Transactions on Information Theory* 46(3), 737–754 (2000)
11. Larsson, N.J., Moffat, A.: Off-line dictionary-based compression. *Proceedings of the IEEE* 88(11), 1722–1732 (2000)
12. Nevill-Manning, C.G., Witten, I.H.: Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research* 7, 67–82 (1997)
13. Pelikan, M.: Analysis of estimation of distribution algorithms and genetic algorithms on NK landscapes. In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pp. 1033–1040. ACM (March 2008)
14. Pelikan, M., Sastry, K., Goldberg, D.E., Butz, M.V., Hauschild, M.: Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pp. 851–858 (2009)
15. Thierens, D.: The linkage tree genetic algorithm. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *PPSN XI. LNCS*, vol. 6238, pp. 264–273. Springer, Heidelberg (2010)

Bent Function Synthesis by Means of Cartesian Genetic Programming

Radek Hrbacek and Vaclav Dvorak

Brno University of Technology,
Faculty of Information Technology
Bozotechnova 2, 61266 Brno, Czech republic
{ihrbacek,dvorak}@fit.vutbr.cz
<http://www.fit.vutbr.cz/~ihrbacek/>

Abstract. In this paper, a new approach to synthesize bent Boolean functions by means of Cartesian Genetic Programming (CGP) is proposed. Bent functions have important applications in cryptography due to their high nonlinearity. However, they are very rare and their discovery using conventional brute force methods is not efficient enough. We show that by using CGP we can routinely design bent functions of up to 16 variables. The evolutionary approach exploits parallelism in both the fitness calculation and the search algorithm.

1 Introduction

Evolutionary Algorithms (EAs) have been recently used in many engineering areas as design and optimization methods. Thanks to the innovation introduced into the design process, they are able to outperform conventional approaches in particular problems. Several types of EAs have been successfully employed in the task of evolutionary circuit design. Besides Genetic Programming (GP) heavily used by John Koza [1] to automatically design analog circuits, regulators, optical systems or antennas, excellent results have been achieved with the use of Cartesian Genetic Programming (CGP) [2]. The applications of CGP include combinational circuits design [3] and optimization [4], digital image filter design [5,6], artificial neural networks design [7] and many others.

The evolutionary design is often very computationally demanding approach. In order to reduce the design time, various application specific accelerators as well as evolutionary algorithm modifications have been proposed. While the former case typically involves parallel fitness function implementation based on FPGA accelerators [6,8] or running on multicore CPUs, GPUs [9] or even computer clusters [3] and exploiting parallelism at various levels (instruction, data, thread or process), the latter one includes genotype representation or search algorithm modifications. In the past, spatially structured evolutionary algorithms have been intensively studied and variety of approaches differing in the used evolutionary algorithm or communication topology has emerged [10,11,12].

While the use of computers and communication networks is becoming more and more popular, one has to seriously deal with the security of the data being

stored or transferred. In cryptography, the two most fundamental techniques to achieve security in systems are *confusion* and *diffusion* [13]. Confusion refers to making a complex relationship between the ciphertext and the key. Thanks to diffusion, the statistical structure of the plain text is dissipated over significant part of the ciphertext, which prevents from reconstructing the original statistical information. In real cryptographic systems, the cipher key is much shorter than the message being encrypted and thus the key has to be reused in some way, often by applying a Boolean function to the key all over again. To avoid decryption by an attacker, the key sequence has to be random. If the Boolean function used to generate the key stream is close to linear, the message can be possibly deciphered. By using functions that are as far from linear as possible, one can build more secure cryptographic systems [14]. These functions, called *bent* functions, are very rare.

The state of the art methods for finding them operate usually on the brute force principle although exploiting some properties of the functions in order to reduce the size of the search space [15]. The number of Boolean functions grows exponentially with the number of variables, while the relative frequency of bent functions decreases. Therefore, for higher number of variables (the literature reports only functions of no more than 8 variables), these methods are not efficient enough. Another approach based on genetic algorithm (GA) is very limited as well. Even though the GA seems to be suitable for this purpose, the proposed approach is not scalable enough [16].

Inspired by the evolutionary design of combinational circuits by means of CGP, we propose a CGP based synthesis of bent Boolean functions. The parallelism at the data, thread and process level has been applied in order to take advantage of modern processor architectures and computer clusters. The scalability of this approach has been earlier verified in the task of evolutionary design of combinational adders and multipliers [3].

The paper is organized as follows. Section 2 introduces bent Boolean functions from the mathematical perspective. CGP is discussed in Section 3 and the proposed evolutionary approach to synthesize bent functions is described in Section 4. Section 5 is dedicated to experiments and the achieved results, final conclusions can be found in Section 6.

2 Bent Boolean Functions

Boolean functions are of great importance for various cryptographic algorithms. Special attention is paid to the design of nonlinear Boolean functions due to their resistance to linear cryptanalysis [17]. This section presents necessary mathematical definitions for the purpose of introduction of bent functions [14,15].

Definition 1. A *Boolean function* is a function of the form $f : D^n \rightarrow D$, where $D = \{0, 1\}$ is a Boolean domain and $n \geq 0$ is the arity of the function. For a function f , let $f_0 = f(0, 0, \dots, 0)$, $f_1 = f(0, 0, \dots, 1)$, ..., $f_{2^n-1} = f(1, 1, \dots, 1)$. $\text{TT}_f = (f_{2^n-1} \dots f_1 f_0)$ is the **truth table representation** of the function f .

Definition 2. A *linear* (Boolean) function is either the constant 0 function or the exclusive OR (XOR) of one or more variables. An *affine* (Boolean) function is a linear function or the complement of a linear function.

Definition 3. The *Hamming distance* $d(f, g)$ between two functions f and g is the number of truth table entries with different values.

Definition 4. The *nonlinearity* NL_f of a function f is the minimum Hamming distance between the function f and an affine function.

Definition 5. Let f be a Boolean function of even arity n , f is a *bent function* iff its nonlinearity NL_f is maximum among n -variable functions.

Affine functions are not suitable for the use in cryptography, since they are susceptible to a linear attack. Therefore, one seeks functions that are as far away (in the Hamming distance) as possible from all the affine functions – these are the bent functions. The nonlinearity of a bent function f of n variables is $NL_f = 2^{n-1} - 2^{\frac{n}{2}-1}$ [18]. This constraint is not applicable for functions of odd

Table 1. Examples of 4-variable Boolean functions and their nonlinearities

	function f	truth table TT_f	nonlinearity NL_f
linear	0	0000000000000000	0
	x_0	1010101010101010	0
	x_1	1100110011001100	0
	$x_1 \oplus x_0$	0110011001100110	0
	x_2	1111000011110000	0
	$x_2 \oplus x_0$	0101101001011010	0
	$x_2 \oplus x_1$	0011110000111100	0
	$x_2 \oplus x_1 \oplus x_0$	1001011010010110	0
	x_3	1111111100000000	0
	$x_3 \oplus x_0$	0101010110101010	0
	$x_3 \oplus x_1$	0011001111001100	0
	$x_3 \oplus x_1 \oplus x_0$	1001100101100110	0
	$x_3 \oplus x_2$	0000111111110000	0
	$x_3 \oplus x_2 \oplus x_0$	1010010101011010	0
$x_3 \oplus x_2 \oplus x_1$	1100001100111100	0	
$x_3 \oplus x_2 \oplus x_1 \oplus x_0$	0110100110010110	0	
nonlinear	x_3x_0	1010101000000000	4
	$x_2x_1x_1 \oplus x_3 \oplus x_0$	1101010100101010	2
	$x_3x_0 \oplus x_1$	0110011011001100	4
	$x_3x_2 \oplus x_1 \oplus x_0$	0110011011001100	4
bent	$x_3x_2 \oplus x_1x_0$	0001000100011110	6
	$x_3x_0 \oplus (x_2 \oplus x_0)x_1 \oplus x_2 \oplus x_0$	1011100000010010	6

arity that can, in general, have greater nonlinearity. This paper deals only with functions of even number of variables.

Examples of Boolean functions of 4 variables can be seen in Table 1. In the first 16 rows, all linear functions are listed, followed by several nonlinear and bent functions, the maximum nonlinearity of 4-variable functions is $NL_f = 2^{4-1} - 2^{\frac{4}{2}-1} = 6$.

The number of different Boolean functions grows exponentially with the number of variables: $N_f(n) = 2^{2^n}$. However, the relative frequency of bent functions decreases very fast (see Table 2) and thus, for $n \geq 6$, identifying them is like looking for a needle in a haystack.

Table 2. Relative frequency of n -variable bent functions [14]

variables n	2	4	6	8
Boolean functions	2^4	2^{16}	2^{64}	2^{256}
bent functions	2^3	$\approx 2^{9.8}$	$\approx 2^{32.3}$	$\approx 2^{106.3}$
relative frequency	2^{-1}	$\approx 2^{-6.2}$	$\approx 2^{-31.7}$	$\approx 2^{-149.7}$

Recently, various approaches based on the properties of bent functions have been proposed, effectively reducing the number of the Boolean functions needed to be verified in order to identify bent functions by means of a brute force search [16,15]. In some special cases, bent functions can be constructed directly [17].

3 Cartesian Genetic Programming

Cartesian genetic programming - a branch of genetic programming - has been introduced by Miller [2] and since then it has been successfully applied to a number of challenging real-world problems [19]. In contrast with GP which uses tree representation, an individual in CGP is represented by a directed acyclic graph. This dissimilarity enables the candidate solution to automatically reuse intermediate results and have multiple outputs, which makes CGP very suitable for design of various kinds of digital circuits, digital filters, etc.

A candidate program in CGP consist of the cartesian grid of $n_r \times n_c$ programmable nodes interconnected by a feed-forward network, as it can be seen in Figure 1. Node inputs can be connected either to one of n_i primary inputs or to a node in preceding l columns, each node has usually a fixed number of inputs $n_{ni} = 2$. Each node can perform one of n_{ni} -input functions from the set Γ . Each of n_o primary circuit outputs is connected either to a primary input or a node output, the output connectivity can be additionally restricted by the o -back parameter. By changing the grid size and the l -back parameter, one can control the area and delay of the circuit.

Thanks to the fixed topology of CGP programs, each chromosome can be encoded using an fixed-sized array of $n_r \cdot n_c \cdot (n_{ni} + 1) + n_o$ integers (n_{ni} inputs

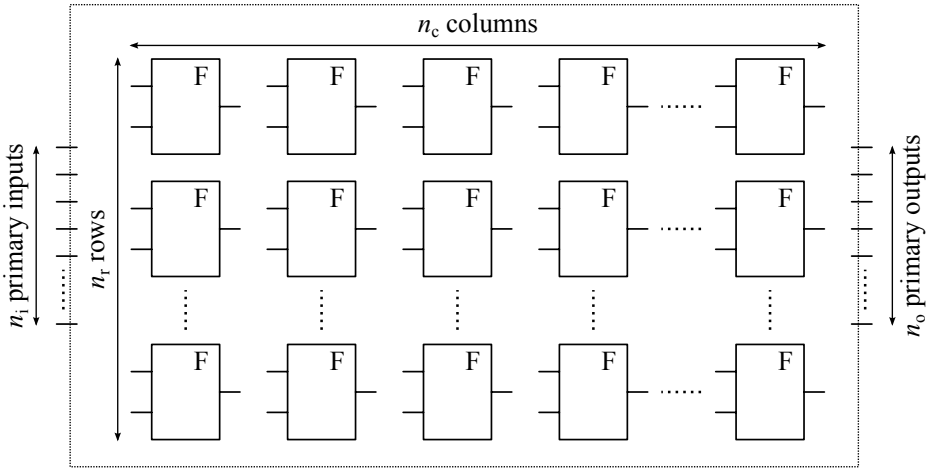


Fig. 1. Cartesian genetic programming scheme

and one function per each node). Each primary input is assigned a number from $\{0, \dots, n_i - 1\}$ and the nodes are assigned numbers from $\{n_i, \dots, n_i + n_r \cdot n_c - 1\}$. Unlike the genotype, the phenotype is of variable length depending on the number of inactive nodes (i.e. nodes whose output is not used by any other node or primary output), which implies the existence of individuals with different genotypes but the same phenotypes. The existence of individuals with different genotypes but with the same fitness value is usually referred to as neutrality. For certain problems, the neutrality significantly reduces the computational effort and helps to find more innovative solutions [20].

CGP uses a simple mutation based $(1 + \lambda)$ evolutionary strategy as a search mechanism, the population size $1 + \lambda$ is mostly very small, typically, λ is between 1 and 15. The initial population is constructed randomly in most cases, however, it can be seeded with a known solution as well (evolutionary optimization) [4]. In each generation, the best individual or a sibling with the same fitness value is passed to the next generation unmodified along with its λ offspring individuals created by means of point mutation operator. The mutation rate m is usually set to modify up to 5% randomly selected genes. Usually, no crossover operator is used in CGP, however, for particular problems (e.g. symbolic regression), special crossover operators have been investigated [21]. None of them has been confirmed as useful for other problem classes so far.

In the case of combinational circuit design, the fitness function is given by the number of correct output bits compared to a specified truth table. All combinations of input values (2^{n_i} test vectors for a circuit with n_i inputs and n_o outputs) have to be fetched to the primary inputs in order to obtain a fully working circuit. $n_o \cdot 2^{n_i}$ output bits have to be verified so as to compute the fitness value.

4 Bent Function Synthesis by Means of CGP

The principle of bent function synthesis by means of CGP is very similar to the case of combinational circuit design, since every Boolean function can be implemented by a combinational circuit. The difference lies in the fitness function. Unlike combinational circuits having fitness value equal to the total number of wrong output bits, the fitness value of a bent function candidate is its nonlinearity, i.e. the lowest Hamming distance from a linear function. Despite the fact, that bent Boolean functions have single output comparing to combinational circuits having arbitrary many outputs, the fitness calculation is computationally more intensive, since the number of linear functions being compared with the candidate individual grows exponentially with the number of variables.

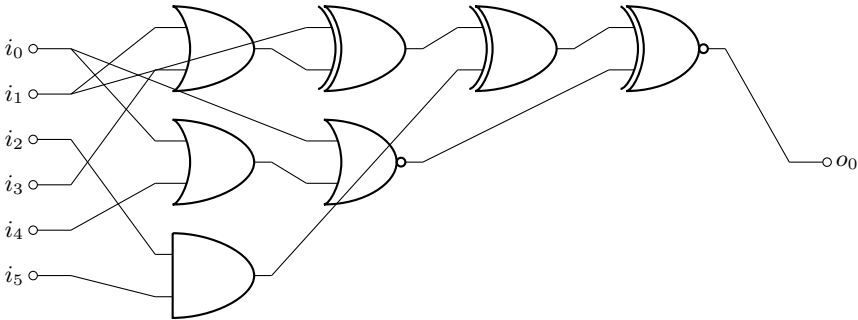


Fig. 2. Example of an CGP individual representing the Boolean function $f(i_5, \dots, i_0) = o_0 = \underline{\underline{((i_1 \oplus (i_1 + i_3)) \oplus i_2 i_5) \oplus i_0 + (i_0 + i_4)}}$ with the truth table $TT_f = 001111000110100100110011011100110111000010100101111111110101010$. This function has nonlinearity $NL_f = 28$ and thus it is bent.

Figure 2 depicts an example of an CGP individual representing a Boolean function. Note that the representation is not optimal in terms of area or delay, since the only significant property is the truth table.

While evaluating an individual’s fitness value, all active genes of the chromosome need to be traversed and their output values need to be calculated. The single output is then compared against all linear functions simply by XORing the values and counting the number of ones. There is no need to compare the values to the remaining affine functions (the complements of linear functions), since the following always holds true:

$$d(f, g) + d(f, g_c) = 2^n, \tag{1}$$

where f, g are arbitrary n -variable Boolean functions and g_c is complementary to g .

The entire evolutionary design process can be accelerated in the same way as it has been done in the case of combinational circuits [3]. The test vectors

can be fed to the CGP individual in parallel, from 64 test vectors within a standard x86-64 register up to 256 test vectors using AVX extension. Moreover, the population can be split over a number of threads, each thread handling a portion of the population. Nevertheless, the number of threads is substantially limited by the population size, which is usually very small in CGP. In order to take advantage of multicore processors or even computer clusters, additional level of parallelism has to be exploited. By introducing spatially structured EA principle, one can scale the evolutionary process onto arbitrary sized computer cluster. Unfortunately, the absence of crossover operator in CGP is a very limiting factor, since most parallel algorithms are based on combining genotypes from different spatially isolated populations. Thus, simple isolated islands model with periodical exchange of the best individual is used [3].

5 Experimental Results

In this section, experiments regarding the ability of the proposed approach to synthesize bent functions are presented. All the experiments were performed on a computer cluster of 112 nodes with the following hardware configuration: 2×8 -core Intel E5-2670, 128 GB RAM, 2×600 GB 15 k scratch hard disks, connected by gigabit Ethernet and Infiniband links.

The performance of the CGP based approach has been examined in terms of the evolution time. The CGP parameters were set as follows on the basis of previous experiments with combinational circuits [3]: the functions set $F = \{\text{BUF, NOT, AND, OR, XOR, NAND, NOR, XNOR}\}$, population of 5 individuals, mutation

Table 3. Bent Boolean functions CGP based synthesis times

n	nodes $n_r \times n_c$	hosts/ threads	time [s]		
			mean	median	std
6	1×50	1/1	0.000819	0.000685	0.000668
8	1×100	1/1	0.00470	0.00343	0.00410
10	1×150	1/1	0.0602	0.0442	0.0483
12	1×200	1/1	2.0443	1.4057	1.9579
		1/4	1.1291	0.8392	1.0610
		4/1	0.8240	0.6267	0.5405
		40/1	0.3859	0.3618	0.1080
14	1×250	1/1	133.202	91.765	146.839
		1/4	76.040	54.954	72.808
		4/1	44.680	35.700	34.165
		40/1	15.806	15.255	4.853
16	1×300	1/1	6223.66	4666.82	4734.02
		1/4	3880.06	3744.23	2571.49
		4/1	1855.79	1543.12	1329.10
		40/1	636.13	565.68	229.06

rate 5%. The number of rows was set to $n_r = 1$ and the l -back parameter was maximal, enabling the greatest connectivity (there were no requirements on the propagation delay). The size of the grid was empirically chosen for each variable count n as a optimal choice with respect to the evolution time (about $10\times$ larger than the average individual found). No limitations on the number of generations were imposed, each run was successful. The spatially structured implementations exchanged the best individual over all populations every 100 generations.

The achieved results can be seen in Table 3, four different configurations of the algorithm were compared – basic single threaded variant, accelerated 4-thread parallel version, 4-island and 40-island spatially structured variants. For each configuration, 100 independent runs were performed and common statistical metrics were calculated – the mean time, the median value and the standard deviation. The evolution times for functions of less than 12 variables are negligible and cannot be improved by means of thread or process level parallelism, because there is not enough work to distribute. For higher numbers of variables, the computational effort grows rapidly and the parallel implementations help significantly to reduce the evolution time. For example, the design of 16-variable bent functions can be sped up $10\times$ on the computer cluster in comparison with the basic single threaded implementation. It shows that even a small number of isolated populations can more efficiently utilize the power of a multicore processor than the multithreaded single population approach. Not only the mean and median times, but also the standard deviations of the evolution times are lower, increasing the probability of finding a bent function in a limited time.

An example of a bent Boolean function of 16 variables synthesized by means of CGP can be seen in Figure 3. Its nonlinearity is 32,640 and the CGP representation has 24 active nodes with the maximum delay of 7.

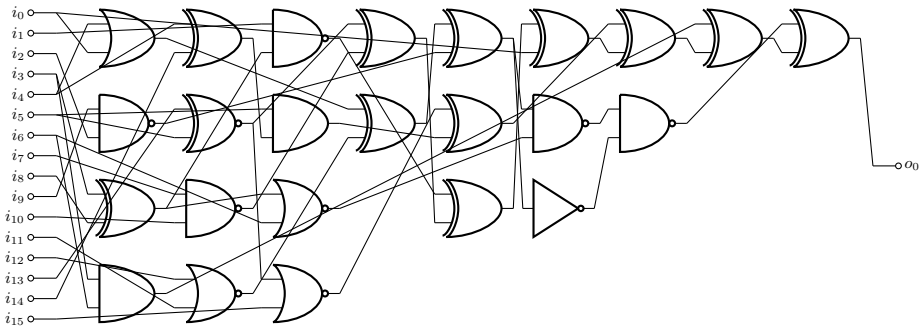


Fig. 3. CGP representation of a 16-variable bent Boolean function

6 Conclusions

In this paper, a new approach to synthesize bent Boolean functions based on CGP has been proposed. Bent functions have applications in cryptography due

to their significant properties – when used in a cipher, their nonlinearity makes cryptanalysis harder. The relative frequency of bent functions among all Boolean functions of the same arity is rapidly decreasing with the number of variables and designing such functions is harder and harder.

It was shown, that by using CGP, we are able to routinely design bent Boolean functions of up to 16 variables. The evolutionary process was sped up by employing various levels of parallelism in both fitness calculation and the search algorithm, which gives a great scalability to the proposed approach. Several algorithm configurations were experimentally compared and it was shown, that by using a simple isolated island model, one can significantly reduce the evolution time. Additional effort has to be made in order to investigate potential common features shared by bent functions found using independent CGP runs.

Even though bent Boolean functions themselves have great properties, in order to achieve maximum confusion in real cryptographic systems, there should be a balance between bits that are changed and that are not. This can be achieved by using *balanced* functions; however, no bent function is balanced and thus a trade-off between nonlinearity and balance has to be sought [17,14]. In our future research, we want to focus on designing such functions by means of evolutionary algorithms. Further work will be also devoted to the optimization of the synthesized functions in terms of area and delay inspired by fast SAT-based optimization methods for complex combinational circuits [4].

Acknowledgements. This work was supported by the Czech Science Foundation project 14-04197S. The access to the CERIT-SC computing and storage facilities provided under the programme Center CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, reg. no. CZ. 1.05/3.2.00/08.0144 is appreciated.

References

1. Koza, J.R.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, Norwell (2003)
2. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
3. Hrbacek, R., Sekanina, L.: Towards highly optimized cartesian genetic programming: From sequential via simd and thread to massive parallel implementation. In: Proceeding of Genetic and Evolutionary Computation Conference, GECCO 2014, Association for Computing Machinery (to appear, 2014)
4. Vasicek, Z., Sekanina, L.: On area minimization of complex combinational circuits using cartesian genetic programming. In: 2012 IEEE World Congress on Computational Intelligence, Institute of Electrical and Electronics Engineers, pp. 2379–2386 (2012)
5. Vasicek, Z., Bidlo, M.: Evolutionary design of robust noise-specific image filters. In: 2011 IEEE Congress on Evolutionary Computation, pp. 269–276. IEEE Computer Society (2011)

6. Hrbacek, R., Sikulova, M.: Coevolutionary cartesian genetic programming in fpga. In: *Advances in Artificial Life, ECAL 2013, Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems*, pp. 431–438. MIT Press (2013)
7. Khan, G., Miller, J.: The cgp developmental network. In: Miller, J.F. (ed.) *Cartesian Genetic Programming. Natural Computing Series*, pp. 255–291. Springer, Heidelberg (2011)
8. Vasicek, Z., Sekanina, L.: Hardware accelerator of cartesian genetic programming with multiple fitness units. *Computing and Informatics* 29(6), 1359–1371 (2010)
9. Harding, S., Banzhaf, W.: Hardware acceleration for cgp: Graphics processing units. In: Miller, J.F. (ed.) *Cartesian Genetic Programming. Natural Computing Series*, pp. 231–253. Springer, Heidelberg (2011)
10. Cantu-Paz, E.: *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Norwell (2000)
11. Tomassini, M.: *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus (2005)
12. Jaros, J.: Multi-gpu island-based genetic algorithm solving the knapsack problem. In: *2012 IEEE World Congress on Computational Intelligence*, pp. 217–224. Institute of Electrical and Electronics Engineers (2012)
13. Shannon, C.: Communication theory of secrecy systems. *Bell System Technical Journal* 28, 656–715 (1949)
14. Butler, J.T., Sasao, T.: Logic functions for cryptography - a tutorial. In: *Proceedings of the Reed-Muller Workshop* (2009)
15. Shafer, J.L., Schneider, S.W., Butler, J.T., Stanica, P.: Enumeration of bent boolean functions by reconfigurable computer. In: Sass, R., Tessier, R. (eds.) *FCCM*, pp. 265–272. IEEE Computer Society (2010)
16. Schneider, S.W.: Finding bent functions using genetic algorithms. Master’s thesis, Naval Postgraduate School, Monterey (2009)
17. Dobbertin, H.: Construction of bent functions and balanced boolean functions with high nonlinearity. In: Preneel, B. (ed.) *FSE 1994. LNCS*, vol. 1008, pp. 61–74. Springer, Heidelberg (1995)
18. Rothaus, O.: On “bent” functions. *Journal of Combinatorial Theory, Series A* 20(3), 300 (1976)
19. Miller, J.F. (ed.): *Cartesian Genetic Programming. Natural Computing Series*. Springer (2011)
20. Miller, J.F., Smith, S.L.: Redundancy and computational efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2), 167–174 (2006)
21. Clegg, J., Walker, J.A., Miller, J.F.: A new crossover technique for cartesian genetic programming. In: *GECCO 2007: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, July 7-11, vol. 2*, pp. 1580–1587. ACM Press, London (2007)

Population Exploration on Genotype Networks in Genetic Programming

Ting Hu¹, Wolfgang Banzhaf², and Jason H. Moore¹

¹ Computational Genetics Laboratory, Geisel School of Medicine, Dartmouth College,
Lebanon, NH 03756, USA

{ting.hu, jason.h.moore}@dartmouth.edu

² Department of Computer Science, Memorial University,
St. John's, NL, A1B 3X5, Canada
banzhaf@mun.ca

Abstract. Redundant genotype-to-phenotype mappings are pervasive in evolutionary computation. Such redundancy allows populations to expand in neutral genotypic regions where mutations to a genotype do not alter the phenotypic outcome. Genotype networks have been proposed as a useful framework to characterize the distribution of neutrality among genotypes and phenotypes. In this study, we examine a simple Genetic Programming model that has a finite and compact genotype space by characterizing its genotype networks. We study the topology of individual genotype networks underlying unique phenotypes, investigate the genotypic properties as vertices in genotype networks, and discuss the correlation of these network properties with robustness and evolvability. Using GP simulations of a population, we demonstrate how an evolutionary population diffuses on genotype networks.

1 Introduction

A remarkable feature of natural evolutionary systems is how they maintain resilience to constant intrinsic and environmental perturbations while remaining adaptive in the face of survival challenges. Robustness [1, 2] and evolvability [3–5] have been discussed as closely related but somewhat contradictory properties in this context. Essentially, both properties reflect how evolutionary systems respond to changes. Robustness enables them to remain intact in the face of deleterious changes, whereas evolvability allows them to innovate to better fit the survival pressures of the environment. Redundancy is a crucial mechanism contributing to both robustness and evolvability. A redundant mapping from multiple genotypes to a phenotype allows genetic variants to expand in neutral mutational spaces. These neutral spaces are genotypic regions in which mutations do not change the phenotype or fitness. Neutral genetic variations by mutations possess the potential for creating novel phenotypes [6]. They serve as a quantitative staging ground for long-term adaptation and innovation. Such neutrality provides a buffer against deleterious mutational perturbations, and augments evolvability by accumulating genetic variations that might be non-neutral under changes of the environmental context [7–10].

Genotype networks, a.k.a. neutral networks, have been proposed as a useful framework for studying neutrality [11–13]. In such networks, genotypes are represented as vertices, and reversible mutational connections, as in common evolutionary systems, are represented as undirected edges between pairs of genotypes. One genotype network is comprised of all genotypes that encode for the same phenotype. Therefore, within a genotype network, edges denote only neutral point mutations. Different genotype networks, i.e. phenotypes, can also be connected through non-neutral point mutations between genotypes that are phenotypically distinguished. Genotype networks provide a global view of how neutrality is distributed among various phenotypes, and hence become a very useful framework to investigate how redundancy contributes to robustness and evolvability. On one hand, studies have shown that evolutionary search really benefits from expanding neutral regions [14, 15]. On the other hand, some evolutionary systems are found to be constrained by the abundance of neutral mutational variants [16].

A redundant mapping from genotype to phenotype is also pervasive in many Evolutionary Computation (EC) systems, especially in Genetic Programming (GP), where multiple genotypes encode identical phenotypes [17–19]. A single point mutation to a genotype is defined as neutral if it does not alter the phenotype or fitness. Such neutrality is largely contributed by the considerable amount of non-coding regions in GP. Departing from early recognition of these non-coding regions as disadvantageous, later extensive investigations and discussions have been conducted on how to characterize and utilize neutrality in GP [20–22]. The notion of genotype networks has also been adopted in many GP neutrality studies [23, 24]. However, most studies characterizing genotype networks are constrained by the infeasibility of enumerating genotypes due to the infinite genotypic space of common GP systems. In a recent study, a quantitative characterization of mutational robustness and evolvability was performed using a simple Linear GP model, where the entire genotype and phenotype spaces are finite and enumerable [25]. It is reported that robustness and evolvability are correlated in a different way at the genotypic, phenotypic, and fitness levels.

In this study, we adopt the same Linear GP model as in a quantitative study on evolvability and robustness [25] to take advantage of its genotype space being amenable to exhaustive enumeration. We characterize topological properties of individual genotype networks and take a close look at vertex importance of genotypes in the networks and how it correlates with robustness and evolvability. Furthermore, using GP simulations, we investigate how an evolutionary population diffuses on genotype networks and how those movements on the genotype networks are reflected in fitness improvements.

2 Methods

2.1 Problem Instance

We consider a simple Linear GP system on a Boolean search problem as in a previous study [25]. In the LGP representation, an individual (or computer program)

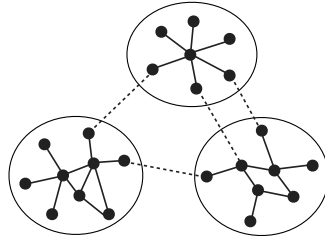


Fig. 1. Schematic diagram of a subset of genotype networks. Each vertex represents a genotype and genotypes encoded to the same phenotype form one genotype network. An edge links two vertices if the two genotypes can be transformed from one to another through a single point mutation. Single point mutations can also connect genotypes from different phenotypes, shown in dashed lines.

consists of a set of L instructions, which are structurally similar to those found in register machine languages. Each instruction has an operator, a set of operands, and a return value. In our study, each instruction consists of an operator drawn from the Boolean function set $\{\text{AND}, \text{OR}, \text{NAND}, \text{NOR}\}$, two Boolean operands, and one Boolean return value. The inputs, operands, and return values are stored in registers with varying read/write permissions. Specifically, R_0 and R_1 are calculation registers that can be read and written, whereas R_2 and R_3 are input registers that are read-only. Thus, a calculation register can serve in an instruction as an operand or a return, but an input register can only be used as an operand. An example program with $L = 3$ is given below.

$$\begin{aligned} R_1 &= R_2 \text{ AND } R_3 \\ R_0 &= R_2 \text{ OR } R_1 \\ R_0 &= R_3 \text{ NAND } R_0 \end{aligned}$$

These instructions are executed sequentially from top to bottom. Prior to program execution, the values of R_0 and R_1 are initialized to **FALSE**. Registers R_2 and R_3 read two Boolean input values. After program execution, the final value in R_0 is returned as output.

2.2 Genotype, Phenotype, and Genotype Networks

We consider each unique LGP program as a *genotype* and the binary Boolean function $f : \mathbf{B}^2 \rightarrow \mathbf{B}$, where $\mathbf{B} = \{\text{TRUE}, \text{FALSE}\}$, represented by the program as its *phenotype*. We set two calculation registers, two input registers and four operators, which means there are $2 \times 4 \times 4 \times 4 = 2^7$ possible instructions and thus 2^{21} possible programs of length $L = 3$. These 2^{21} programs define the finite genotype space mapping to the 16 possible binary Boolean functions $f : \mathbf{B}^2 \rightarrow \mathbf{B}$ as phenotypes.

Genotypes transform from one to another through point mutations. These mutational connections can be well modeled by networks. The framework of

genotype networks has been proposed to study how mutational connections are distributed among genotypes underlying various phenotypes [11–13]. A *genotype network* is comprised of all the genotypes, as vertices, that represent the same phenotype. An edge connects a pair of genotypes if they can be transferred from each other through a single point mutation (see Fig. 1).

Different phenotypes can have varying genotype network properties, and investigating these network properties provides insights into how an evolutionary population explores the genotype space by expanding in genotype networks. We take advantage of the simple yet representative LGP system to fully characterize the entire genotype space by enumerating all genotypes and constructing all 16 genotype networks. Then, for each genotype network, we look at their network properties including network size, i.e. the total number of vertices, network degree distribution and vertex closeness centrality.

The degree of a vertex in a network is the number of its connected neighbors. In the framework of genotype networks, vertex degree reflects how robust a genotype is when subject to point mutations. High degree vertices are genotypes that are more likely to maintain their phenotypes under point mutations. Vertex degree distribution describes the global connectivity of a network. At the vertex level, centrality measures the importance of a vertex in the network. There are a number of centrality measures that capture the individual contribution of vertices to a network. In the current study, we look at the *closeness centrality*, denoted as $\frac{1}{\sum_{j \neq i} d_{ij}}$ of a vertex i , where d_{ij} is the distance, i.e. the shortest path, between vertices i and j [26, 27]. Closeness centrality describes how easily a given vertex can reach all other vertices. A higher closeness centrality indicates a more central position of a vertex in the network. Beyond mutational connections within genotype network, genotypes can mutate into different phenotypes through non-neutral single point mutations. The *evolvability* of a genotype is defined as the number of unique phenotypes that it can reach through single point mutations [13]. This definition is intuitive in that if a genotype is adjacent to genotypes from many other different phenotypes, it is considered more evolvable.

2.3 Population Evolution

Population evolution is simulated to investigate how a population diffuses on genotype networks. The initial population includes $|P|$ randomly chosen genotypes from one given phenotype. Then for each generational iteration, a number of individuals are subject to single point mutations, according to a mutation rate r , and both $|P|$ parents and $r \times |P|$ offspring are competing in a tournament selection to form the next generation of $|P|$ individuals. We set one particular phenotype as the target and let the population evolve towards it. The evolution process is terminated once the entire population converges to the target.

A single point mutation can apply to any locus of a genome, including the return register, one of the two operand registers, or the operation function. The fitness value of a genotype is calculated based on the mutational potential from its phenotype to the target phenotype. Specifically, let v_{ij} denote the total

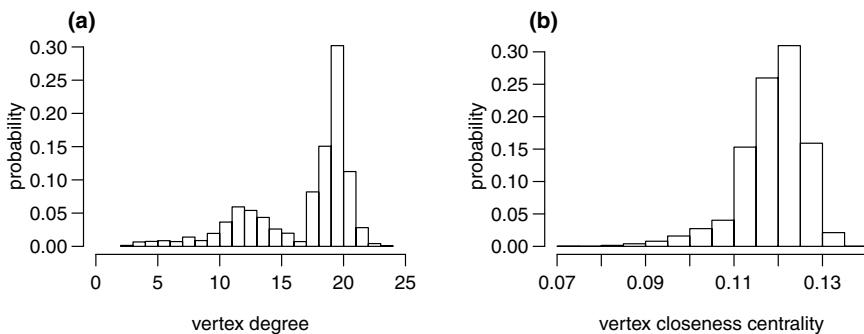


Fig. 2. Genotype network properties of phenotype NAND. (a) Distribution of vertex degree. (b) Distribution of vertex closeness centrality.

number of possible single point mutations that can transform genotypes from phenotype i to phenotype j . The fitness of genotypes from phenotype k with regard to the target phenotype t is defined as $f_t(k) = \frac{v_{kt}}{\sum_{j \neq k} v_{kj}}$. This fitness calculation is defined following the intuition that a phenotype with a higher mutational potential towards the target is rewarded with a higher fitness value.

3 Results and Discussion

3.1 Properties of Genotype Networks

For our particular LGP system, the distribution of genotypes among different phenotypes is highly heterogeneous. The size of genotype networks ranges from a minimum of 64 genotypes (for phenotypes EQUAL and XOR) to a maximum of 617,024 genotypes (for FALSE), occupying between $\ll 0.1\%$ and 29.4% of the entire genotype space, respectively. The mutational connections among phenotypes are also unevenly distributed. Out of the total 16 genotype networks, 10 are mutationally accessible from all other genotype networks, 4 are adjacent to 14 other phenotypes, and the two smallest genotype networks (EQUAL and XOR) have 13 phenotype neighbors. Moreover, these two smallest genotype networks are comprised of 64 individual islands, i.e. all 64 genotypes mutate away from their phenotype with any single point mutations, whereas the other 14 genotype networks contain single connected components.

Due to the symmetry of Boolean functions, some genotype networks share the same topological properties, e.g. phenotypes $x \geq y$ and $x \leq y$. Interestingly, all genotype networks, excluding EQUAL and XOR that have all genotypes as isolated vertices, share the bi-modal vertex degree distribution. Fig. 2(a) shows the degree distribution of the representative genotype network NAND. This degree distribution suggests that the genotype networks are comprised of a dense core of highly connected genotypes, as well as a cluster of genotypes towards the periphery. The vertex closeness centrality has a uni-modal distribution (Fig. 2(b)),

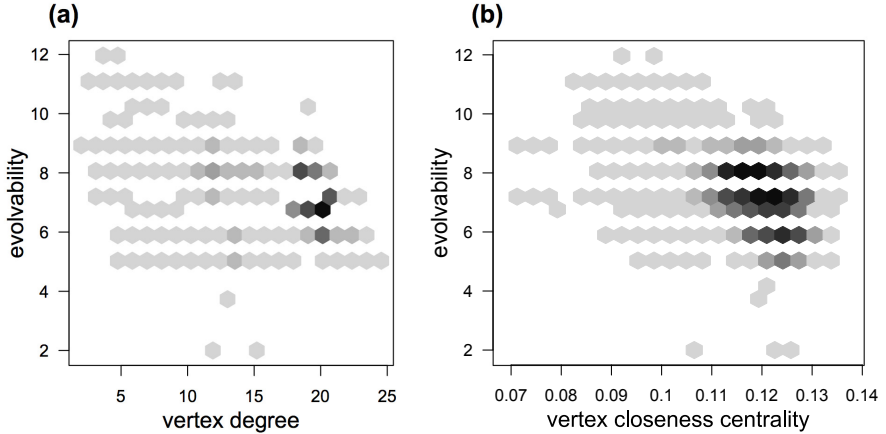


Fig. 3. The correlations of genotypic evolvability and (a) vertex degree and (b) vertex closeness centrality of the NAND genotype network. Grayscale of hexagons indicates the density of value intervals.

suggesting that most vertices are about equally accessible from other vertices in the network. The vertex degree and closeness centrality are also positively correlated (Spearman’s rank correlation $\rho = 0.5417$, $p < 2 \times 10^{-16}$).

Recall that the evolvability of a given genotype is measured as the number of accessible unique phenotypes through single point mutations. We then look into how genotypic evolvability correlates with the degree and closeness centrality of a genotype in the network. Fig. 3 shows evolvability as a function of (a) vertex degree and (b) vertex closeness centrality. It can be observed that both the vertex degree and closeness centrality are negatively correlated with evolvability (Spearman’s rank correlations $\rho = -0.3379$, $p < 2 \times 10^{-16}$ and $\rho = -0.3865$, $p < 2 \times 10^{-16}$, respectively). This suggests that the dense center cores of genotype networks have less access to other unique phenotypes, i.e. are less evolvable, than the genotypes at the periphery.

3.2 Population Diffusion on Genotype Networks

After quantifying the static properties of genotype networks, we now use population evolution to investigate how a population diffuses on genotype networks. We set one of the least representative phenotypes, $\mathbf{x} = \mathbf{y}$, as the evolution target to allow evolution to proceed for a longer time. $|P| = 500$ individuals of a given starting phenotype are randomly sampled as the initial population. We set mutation rate $r = 0.1$ and use a tournament selection of size two. For each starting phenotype configuration we collect 1,000 independent runs, and for each run the required number of generations that a population converges to the target phenotype is recorded.

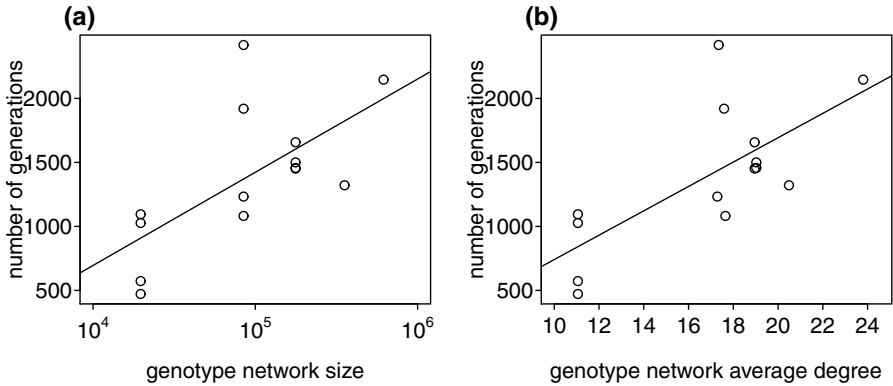


Fig. 4. The required evolution time as a function of the starting phenotype’s genotype network properties. (a) The number of required generations increases as the genotype network becomes larger. (b) Evolution also requires a longer time if the starting phenotype’s average vertex degree is larger. The lines provide a visual guide of their correlations.

Fig. 4 shows the correlations of the required evolution time and the starting phenotype’s properties. A population needs a longer time to reach and converge to the target if it starts from a larger genotype network (Fig. 4(a), Spearman’s rank correlation $\rho = 0.6640$, $p = 0.0096$). This positive correlation also exists between the evolution time and the starting phenotype’s average vertex degree (Fig. 4(b), Spearman’s rank correlation $\rho = 0.6326$, $p = 0.0152$). This suggests that it takes a population longer to evolve if it starts from a larger and more connected genotype network. This finding contradicts Wagner’s RNA results where larger phenotypes, i.e. more robust phenotypes, are more evolvable [13], but agrees with Cowperthwaite’s argument that the abundance of genotype networks constrains evolution [16]. We would like to point out that their correlation crucially depends on the properties of an evolutionary system, specifically how the genotype networks are adjacent to each other globally and where the target phenotype is located. For our LGP system, the target phenotype can be accessed from 13 other phenotypes, such that there are many possible paths to find the target. Therefore, evolution is expected to take a longer time moving out of large genotype networks and exploring novel phenotypes.

Last we take a close look at how a population diffuses on genotype networks as evolution proceeds. Fig. 5 shows the average vertex degree of a population changes as a function of generation in a typical run. The population starts with the average vertex degree of the starting genotype network NAND. Individuals then quickly move towards the periphery of the networks (generation 1 to 10). During the subsequent search, the population visits many other phenotypes, but leaves without going into their center cores (generation 11 to 160). The population reaches the first genotype of the target phenotype at generation 161, and quickly converges to the target in the next 33 generations. Also note that,

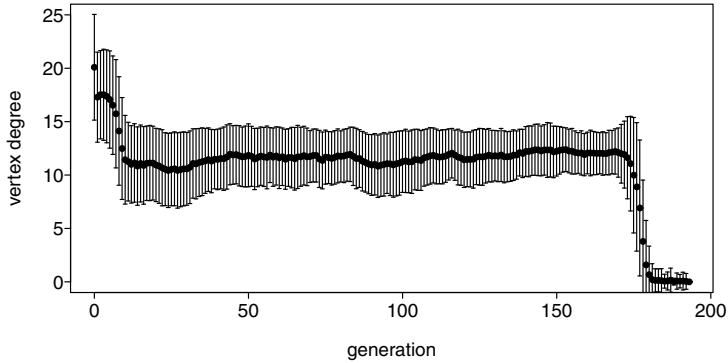


Fig. 5. The change of average vertex degree of an evolving population as evolution proceeds in a typical run of starting phenotype NAND. Points are population mean and error bars are standard deviations.

data not shown here, the change of vertex closeness centrality follows the same trend as the vertex degree since they are positively correlated.

4 Concluding Remarks

Here we have used a simple yet representative LGP system to fully characterize all individual mutational genotype networks by exhaustively enumerating the entire genotype and phenotype spaces. The 16 unique phenotypes are represented by 16 genotype networks that possess both shared and distinguishing properties. The two smallest genotype networks are comprised of isolated individual vertices, whereas all other networks contain single connected components. The connected genotype networks share similar bi-modal degree distributions, which indicate that the networks are comprised of a dense core and a well-connected periphery. In such genotype networks, vertices with high degrees are more likely located in the center of connecting all other vertices. However, these high-degree and high-centrality genotypes are less evolvable towards novel phenotypes.

By simulating population evolution, we find that a population requires more time to find a target if it starts from a larger genotype network. This observation conforms well to the static characterization of genotype properties in networks. We would like to point out that how the abundance of mutational variants contribute to evolvability crucially depends on the distribution of neutrality among various phenotypes and where the target phenotype is located. Our simulation also shows how an evolutionary population diffuses on genotype networks. It moves from the center of a network towards the periphery as the evolutionary search proceeds, accompanied by fitness improvements, and stays on the periphery of genotype networks visited until the target phenotype is reached.

The findings of this study provide insights on how neutrality is distributed in a typical LGP system. We conjecture that genotype networks could be shaped

very differently in other GP systems, however our current observations capture many general properties of GP, and might even be applicable to other EC systems. Specifically, the distribution of neutrality is very heterogenous among various phenotypes. Some genotype networks, i.e. phenotypes, could be orders of magnitude larger than others. Moreover, the mutational connections among phenotypes are biased, where a phenotype has more potential to mutate to particular phenotypes and is less likely to mutate to or is even disconnected from some phenotypes. The success of an innovative evolutionary search crucially depends on locating the target phenotype, i.e. whether it is accessible from many other phenotypes, and on finding an efficient mutational path towards it.

In future studies, we expect to use our methodology in other GP- or EC-systems and test if our observations and conjectures hold for a wider range of applications. It would be helpful to look into how a particular EC representation correlates with genotype network properties, such that we can gain a better understanding of how a representation influences evolutionary search and how we could improve the performance of an evolutionary algorithm by designing more appropriate representations.

Acknowledgments. This work was supported by National Institute of Health (USA) grants R01-LM009012, R01-LM010098, R01-AI59694, P20-GM103506, and P20-GM103534. W.B. acknowledges support from NSERC Discovery Grants, under RGPIN 283304-2012.

References

1. Lenski, R.E., Barrick, J.E., Ofria, C.: Balancing robustness and evolvability. *PLoS Biology* 4(12), e428 (2006)
2. van Nimwegen, E., Crutchfield, J.P., Huynen, M.A.: Neutral evolution of mutational robustness. *Proceedings of the National Academy of Sciences* 96(17), 9716–9720 (1999)
3. Kirschner, M., Gerhart, J.: Evolvability. *Proceedings of the National Academy of Sciences* 95, 8420–8427 (1998)
4. Pigliucci, M.: Is evolvability evolvable? *Nature Review Genetics* 9, 75–82 (2008)
5. Wagner, A.: Robustness, evolvability, and neutrality. *Federation of European Biochemical Societies Letters* 579(8), 1772–1778 (2005)
6. Masel, J., Trotter, M.V.: Robustness and evolvability. *Trends in Genetics* 26, 406–414 (2010)
7. Draghi, J.A., Parsons, T.L., Wagner, G.P., Plotkin, J.B.: Mutational robustness can facilitate adaptation. *Nature* 463, 353–355 (2010)
8. Landry, C.R., Lemos, B., Rifkin, S.A., Dickinson, W.J., Hartl, D.L.: Genetic properties influencing the evolvability of gene expression. *Science* 317, 118–121 (2007)
9. McBride, R.C., Ogbunugafor, C.B., Turner, P.E.: Robustness promotes evolvability of thermotolerance in an RNA virus. *BMC Evolutionary Biology* 8, 231 (2008)
10. de Visser, J.A.G.M., Hermisson, J., Wagner, G.P., Meyers, L.A., Bagheri-Chaichian, H., et al.: Evolution and detection of genetic robustness. *Evolution* 57(9), 1959–1972 (2003)

11. Reidys, C., Stadler, P.F., Schuster, P.: Generic properties of combinatorial maps: neutral networks of RNA secondary structures. *Bulletin of Mathematical Biology* 59(2), 339–397 (1997)
12. Schuster, P., Fontana, W., Stadler, P.F., Hofacker, I.L.: From sequences to shapes and back: A case study in RNA secondary structures. *Proceedings of The Royal Society B* 255, 279–284 (1994)
13. Wagner, A.: Robustness and evolvability: A paradox resolved. *Proceedings of The Royal Society B* 275(1630), 91–100 (2008)
14. Ciliberti, S., Martin, O.C., Wagner, A.: Innovation and robustness in complex regulatory gene networks. *Proceedings of the National Academy of Sciences* 104(34), 13591–13596 (2007)
15. Wilke, C.O.: Adaptive evolution on neutral networks. *Bulletin of Mathematical Biology* 63, 715–730 (2001)
16. Cowperthwaite, M.C., Economo, E.P., Harcombe, W.R., Miller, E.L., Meyers, L.A.: The ascent of the abundant: How mutational networks constrain evolution. *PLoS Computational Biology* 4(7), e1000110 (2008)
17. Banzhaf, W.: Genotype-phenotype mapping and neutral variation - a case study in genetic programming. In: Davidor, Y., Schwefel, H.P., Manner, R. (eds.) *PPSN 1994. LNCS*, vol. 866, pp. 322–332. Springer, Heidelberg (1994)
18. Rothlauf, F., Goldberg, D.E.: Redundant representations in evolutionary computation. *Evolutionary Computation* 11(4), 381–415 (2003)
19. Hu, T., Banzhaf, W.: Evolvability and speed of evolutionary algorithms in light of recent developments in biology. *Journal of Artificial Evolution and Applications* 568375 (2010)
20. Galvan-Lopez, E., Poli, R.: An empirical investigation of how and why neutrality affects evolutionary search. In: Cattolico, M. (ed.) *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1149–1156 (2006)
21. Hu, T., Banzhaf, W.: Neutrality and variability: Two sides of evolvability in linear genetic programming. In: Rothlauf, F. (ed.) *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 963–970 (2009)
22. Soule, T.: Resilient individuals improve evolutionary search. *Artificial Life* 12, 17–34 (2006)
23. Banzhaf, W., Leier, A.: Evolution on neutral networks in genetic programming. In: Yu, T., Riolo, R., Worzel, B. (eds.) *Genetic Programming Theory and Practice III*, pp. 207–221. Springer (2006)
24. Ebner, M., Shackleton, M., Shipman, R.: How neutral networks influence evolvability. *Complexity* 7(2), 19–33 (2002)
25. Hu, T., Payne, J.L., Banzhaf, W., Moore, J.H.: Evolutionary dynamics on multiple scales: A quantitative analysis of the interplay between genotype, phenotype, and fitness in linear genetic programming. *Genetic Programming and Evolvable Machines* 13, 305–337 (2012)
26. Bavelas, A.: Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America* 22, 725–730 (1950)
27. Sabidussi, G.: The centrality index of a graph. *Psychometrika* 31(4), 581–603 (1966)

Improving Genetic Programming with Behavioral Consistency Measure

Krzysztof Krawiec¹ and Armando Solar-Lezama²

¹ Institute of Computing Science, Poznan University of Technology, Poznań, Poland

`krawiec@cs.put.poznan.pl`

² Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology, Cambridge, MA, USA

`asolar@csail.mit.edu`

Abstract. Program synthesis tasks usually specify only the desired output of a program and do not state any expectations about its internal behavior. The intermediate execution states reached by a running program can be nonetheless deemed as more or less preferred according to their information content with respect to the desired output. In this paper, a consistency measure is proposed that implements this observation. When used as an additional search objective in a typical genetic programming setting, this measure improves the success rate on a suite of 35 benchmarks in a statistically significant way.

Keywords: Program synthesis, genetic programming, entropy, multi-objective search.

1 Introduction

One of the main challenges for genetic programming (GP)—or for that matter, for any approach to program synthesis based on explicit search over a space of programs—is to decide on a fitness function that captures the relative quality of different proposed solutions. A common approach is to consider the quality of the *output* of a given program on a set of candidate inputs, possibly augmented with some structural constraints to prevent the search algorithm from overfitting to the training inputs (examples). The problem with this approach is that there is a fundamental mismatch between the search approach, which operates on the *structure* of the program, and the fitness function which is based on its input/output *behavior*. An ideal fitness function would instead be one that rewards programs that are structurally close to a correct solution: a program that is only a few small modifications away from being correct is better than one that has to be completely transformed in order to work, even if the former produces incorrect output on every input. The problem, of course, is that we do not know how the correct program looks like—if we did, we would not be searching for it—so we are left with behavior-based measures of correctness.

In this paper, we propose a new fitness measure that tries to better capture how close a proposed solution is from being correct by assessing the quality of intermediate values of the program in addition to the quality of the output.

At first glance, it may seem that such a measure would run into the same problem as any other structural quality measure: since we do not know what the correct solution looks like, we also do not know what the intermediate values produced in the process of that computation should be. However, we do know that these intermediate values must preserve certain information about the input that is necessary to produce the desired output. Our main contribution is to show that information theoretic measures of the quality of intermediate values can improve upon traditional output-based measures in a statistically significant way.

The intuition for the approach is as follows: if two inputs are supposed to produce different outputs, then any program prefix that loses the information necessary to distinguish between these two inputs is doomed to failure since once lost, the information cannot be recovered. By contrast, if two inputs are supposed to produce the same output, then a program prefix that produces the same value for these two inputs will eliminate the risk that the rest of the program incorrectly assigns different values to these two inputs. The rest of the paper formalizes this intuition and exploits it in a new fitness measure based on *information consistency*. We implement the approach in a single-objective and a multi-objective variant, and evaluate it on 35 benchmarks representing three different domains. The experimental results clearly suggest that the involvement of information consistency almost always increases the likelihood of synthesizing the correct program.

2 Information Consistency

Conceptual Background. The proposed approach works by analyzing the internal behavior of programs, by which we mean the effects of computation conducted at intermediate stages of program execution. Conceptually, we examine such effects by placing *traps* (breakpoints) at selected locations in program code and inspecting the program state when execution reaches these traps. We assume that traps are not embraced by loops or conditional statements, thus ensuring that a deterministic program always visits all traps in the same order *independently of program input*.

When stopped at a trap, an executing program produces a certain *state* of the execution environment, which we assume to depend only on the part of the program that is syntactically associated with the trap (or, informally speaking, ‘scoped’ by the trap). Depending on the adopted programming paradigm, the particular interpretation of these notions will vary. For register-based sequential programs, a state would be the contents of all registers, and it would depend on the entire code executed so far (i.e., program prefix ending at the trap). For purely functional programming, by state we would mean the value calculated in the corresponding function call, caught in the process of being passed to the caller. In this study we consider functional tree-based GP and will place traps at tree nodes, so a state will be the value returned by the corresponding subtree.

Let s_i^k denote the state of the program at the k th trap when executed on the i th training example (input data). The sequence of the states s_i^1, s_i^2, \dots traversed

by a program for a given i th example will be called the *trace* for that example.

Our method is intended to promote programs that reach states that exhibit a certain form of consistency with the desired output. Given two examples for which the desired program outputs are y_i and y_j , two situations are possible:

Case #1: $y_i \neq y_j$. In this case, the program should maintain distinct behaviors for examples i and j . If that is not the case, i.e., the program leads to $s_i^k = s_j^k$ at some k th trap, this should be considered undesired and penalized. It is so because once program traces for two examples merge, they cannot diverge anymore (while they should if distinct y_i and y_j are to be produced at the end of program execution).

Case #2: $y_i = y_j$. In this case, the program should reach the same state for the i th and j th example at some stage of its execution, i.e., it is desirable to observe $s_i^k = s_j^k$. Conversely, a program for which no trap with this property exists is unlikely to end up with correct output, and thus should be penalized.

In other words, the observed execution states should form *equivalence classes* that are *consistent* with the equivalence classes induced by the desired program output. Ideally, a program would feature a trap for which this consistency is full, i.e., $s_i^k = s_j^k \iff y_i = y_j$. Once such a state emerges in an evolving program, producing the correct output is only a matter of one-to-one mapping from intermediate execution states to the final execution states. Although realizing such a mapping can be still difficult in some programming languages, we hypothesize that promoting programs that feature locations with such properties is desirable¹.

Consistency Measure. To promote and demote programs in an evolving population according to how their internal behaviors meet the characteristics discussed above, we design a measure based on information theory to quantitatively assess the consistency of program behavior with the desired output. Let us start from an observation that the process of program execution is usually accompanied by gradual loss of information in the execution environment. More precisely, a deterministic program can at most maintain the amount of information present in the data it has been applied to, but is unable to increase it. In an extreme case, a program that ignores its input and always returns the same output, reduces the amount of information to zero.

In terms of the notions introduced above, information is lost every time the traces associated with particular examples merge, i.e., every time the program, when applied to distinct training examples, reaches the same execution state (more specifically, if $s_i^k \neq s_j^k$ for a certain k th trap, but $s_i^l = s_j^l$ for some subsequent l th ($l > k$) trap). Depending on the particular pair of examples (i, j), that loss may be detrimental (when it increases the likelihood of producing the same output; case #1 in Section 2), or desirable (case #2).

¹ Whether the errors presented in Case #1 and #2 are *critical* depends on the adopted programming paradigm. For sequential programs, a state captures the entirety of computation conducted so far, so the erroneous merging or non-merging of traces cannot be fixed by subsequent execution of program suffix. For functional programming, however, other parts of the program can substitute for such a deficiency.

To assess the amount of lost information, we associate a random variable S_k with the k th trap, where the values of S_k are the states associated with particular examples. Analogously, we define a random variable Y representing the desired output. Based on the concept of conditional entropy ($H(X|Y) = -\sum \Pr(X|Y) \log_2 \Pr(X|Y)$), we consider:

- $H(Y|S_k)$, i.e., the amount of information that Y adds to S_k . In particular, if $H(Y|S_k) > 0$, then S_k alone is not sufficient to predict the value of Y .
- $H(S_k|Y)$, the amount of information that S_k adds to Y . Large values of $H(S_k|Y)$ indicate that S_k partitions the set of examples into many equivalence classes.

In connection to our previous considerations, every time the traces for two or more examples merge between the k th and $(k + 1)$ th trap, either the former term increases ($H(Y|S_k) > H(Y|S_{k+1})$) or the latter term drops ($H(S_k|Y) > H(S_{k+1}|Y)$). Both $H(Y|S_k)$ and $H(S_k|Y)$ attain zero if and only if S_k perfectly ‘correlates’ (is consistent) with Y , i.e., $s_i^k = s_j^k \iff y_i = y_j$.

Following this observation, we base our measure on the sum of the above terms. We define the (minimized) *information consistency* I of a program p according to the trap at which the total two-way conditional entropy attains its minimum, i.e.,

$$I(p) = \min_k H(Y|S_k(p)) + H(S_k(p)|Y) \quad (1)$$

where $S_k(p)$ is the random variable associated with the k th trap set on program p . The lower values of I are more desired as they indicate program behavior that is more consistent with Y . By using the minimum operator for aggregation over program traps, $I(p)$ rewards p for the part of its behavior that is most consistent with the desired output Y , even if otherwise (i.e., at other locations/traps) it behaves in a way that is unrelated to Y . This is intended to promote programs that are partially correct and thus feature code pieces that can prove useful in new programs².

Example. For simplicity, we illustrate these concepts on a linear program, by which we mean a program that reads in the input data only once, at the beginning of its execution and involves no loops or branches.

Consider a programming task defined by five examples, and a linear program with three traps. Table 1 presents the states traversed by the program for particular examples, where for brevity we encode program states in lowercase letters. We use different symbols for particular traps to emphasize that the random variables may assume values from different domains (though in practice, e.g., a , f , and j could represent the same value). The last column of the table presents the desired output of the program.

The table presents also the conditional entropy for consecutive traps. $H(Y|S_k)$ remains at zero for at the first and second trap (S_1 and S_2), and then increases

² Note that the term minimized in (1) can be alternatively expressed as $H(Y, S_k(p)) - I(Y; S_k(p))$, where $H(Y, S_k(p))$ stands for joint entropy and I is the mutual information. However, minimization of (1) is not equivalent to maximization of mutual information only, as $H(Y, S_k(p))$ may also vary from trap to trap.

Table 1. Exemplary calculation of consistency measure for a program with three traps, executed on five examples. The minimum of $H(Y|S_k) + H(S_k|Y)$ over the traps, marked in bold, is the information consistency I of this program.

Example	S_1	S_2	S_3	Y
1	a	f	j	1
2	b	g	k	2
3	c	g	k	2
4	d	h	j	2
5	e	i	j	3
$H(Y S_k)$	0	0	0.95	
$H(S_k Y)$	0.95	0.55	0.55	
$H(Y S_k) + H(S_k Y)$	0.95	0.55	1.50	

for S_3 . As the states in consecutive traps merge (e.g., b and c in S_1 merge into g in S_2), the corresponding random variables S_k carry less and less information, and the entropy of Y conditioned on S_k grows.

Conversely, $H(S_k|Y)$ cannot grow with consecutive traps, because the collapsing states reduce the amount of information. In an ideal case, $H(Y|S_k) = H(S_k|Y) = 0$, i.e., neither the state adds any information to the desired output, nor the reverse. This would happen if g and h collapsed into a single state in S_2 , which would then be perfectly correlated with Y . \square

The motivations for our initial assumption that no trap is inside a loop or conditional statement should become clear now. Otherwise, a program could, depending on the input data, visit some traps more than once, not visit some traps, or visit them in a different order. In other words, the execution traces could not be ‘aligned’ in a reasonable way. This in turn would make it impossible to compare the corresponding execution states in a meaningful manner.

Related Work. In inspecting the internal behavior of programs, consistency measure relates to our former work on *behavioral search drivers* for GP [5]. However, the methods presented there were after a more general class of behaviors, and used machine learning inducers to identify them. Compared to them, here we focus exclusively on information contents, which allows us to assess the impact of this specific aspect of program behavior on search efficiency.

In a broader context, studying the internal behavior of programs can be seen as an extension of *semantic GP*, a new branch in GP research initiated by McPhee *et al.* [8]. However, most of work conducted in this area, including recent work (see, e.g., [9]), takes into account only the final output of programs.

The requirement that a program prefix should not lose information necessary to distinguish inputs that must lead to distinct outputs has been used in both constraint-based software synthesis [10] and interpolant-based hardware synthesis [3]. In different contexts, both of those works use the constraint to ensure that a given prefix can be completed to be equivalent to a given program.

Finally, the consistency measure proposed above will be integrated into evolutionary workflow, which can be done, among others, by treating it as an additional search objective. This can be considered as a form of *multiobjectivization* by Knowles *et al.* [4], meant as extending the original problem formulation by helper objectives intended to make problem solving more efficient.

3 Experiment

In the following, we examine the usefulness of our consistency measure in the framework of tree-based GP, with the programs being expression trees that feature no loops nor conditional statements, and have no side effects. A trap placed at a tree node will allow us to examine the value calculated by the program subtree (subprogram) rooted at that node. The state will be simply the value calculated by that subtree.

An important consequence of associating execution states with program subtrees is that a state does not capture the *entirety* of the computation conducted so far by a program, but only a *local* execution state (depending on the corresponding subtree; cf. Footnote 1). This however does not undermine the rationale presented in Section 2: a subtree that behaves consistently with the desired program outcome is a potentially useful piece of code, and a program that hosts it should be promoted. However, contrary to strictly linear programs, even if a subtree merges two or more states that should not be merged (and leads to $H(Y|S_k) > 0$), a program is not necessarily destined to perform bad, as other, independent program subtrees can still be able to distinguish those states.

Configurations. The goal of the experiment is to assess the impact on the information elicited by the consistency measure from the evolving programs. We consider two ways of integrating $I(p)$ (Formula (1)) into the conventional GP workflow: by redefining the scalar fitness in the conventional single-objective evolutionary workflow, and by using $I(p)$ as an additional objective in a multi-objective setting. In the former setup, called **FxI** in the following, the (minimized) program fitness is defined as

$$(1 + F(p))(1 + I(p)) \quad (2)$$

where $F(p)$ is the conventional program error (Hamming or city-block distance, depending on the domain). For the sake of simplicity, we deliberately abstain from exploiting the trade-off between F and I via weighing.

For the multiobjective setup (**FI** in the following) we assign a two-dimensional fitness $(F(p), I(p))$ to a program and employ the Non-Dominated Sorting Genetic Algorithm II (NSGA-II, [1]) at the selection stage, the arguably most popular method of multiobjective evolutionary optimization.

The baseline for the above methods is the conventional Koza-style GP (**F** in the following), which uses $F(p)$ as the only search objective.

To avoid making the (potentially biased) decisions where to set traps in programs, we stop program execution and calculate the two-way entropy (Eq. 1) after *every single instruction*. This imposes substantial computational burden on the evaluation process, which we take into account in one of the experiments.

A run is terminated when an ideal solution is found ($F = 0$) or the maximum number of 250 generations has elapsed. The percentage of runs that ended with the former result (out of the total of 30 independent evolutionary runs) forms the *success rate*, the performance metric we use in the following. A total of 30×35 benchmarks \times 6 configurations = 6300 runs has been conducted.

Table 2. The benchmarks. v – number of input variables, m – number of tests, k – number of semantically unique programs.

Domain	Instruction set	Problem	v	m	k
Boolean	and, nand, or, nor	Cmp6, Maj6, Mux6, Par6	6	64	2^{64}
		Cmp8, Maj8, Par8	8	256	2^{256}
		Mux11	11	2048	2^{2048}
Categorical	$a_l(x, y)$	D-a1, D-a2, D-a3, D-a4, D-a5	3	27	3^{27}
	$a_l(x, y)$	M-a1, M-a2, M-a3, M-a4, M-a5	3	15	3^{15}
Regression	+, −, *, %, sin, cos, log, exp, −x	Keij1, Keij4, Nguy3..7, Sext	1		
		Keij5, Keij11..14, Nguy9..10, Nguy12	2	20	–
		Keij15	3		

For single-objective configurations, tournament of size 5 is used for selection. Except for the elements of the setup that have to differ across domains because of their different natures, all benchmarks in all considered domains use the same parameter settings. The remaining parameters use ECJ’s defaults [6].

Program Synthesis Problems. Table 2 presents the 35 benchmark problems used in our experiment, which come from three domains: Boolean (8 benchmarks), categorical (10 benchmarks), and regression (17 benchmarks). Table 2 summarizes the problems, listing the instruction set, the number of variables, tests, and the cardinality of the search space (where countable). Note that none of the instruction sets contains constants.

The particular **Boolean problems** are defined as follows. For an v -bit comparator Cmp v , a program is required to return *true* if the $\frac{v}{2}$ least significant input bits encode a number that is smaller than the number represented by the $\frac{v}{2}$ most significant bits. In case of the majority Maj v problems, *true* should be returned if more than half of the input variables are *true*. For the multiplexer Mul v , the state of the addressed input should be returned (6-bit multiplexer uses two inputs to address the remaining four inputs, 11-bit multiplexer uses three inputs to address the remaining eight inputs). In the parity Par v problems, *true* should be returned if and only if an odd number of *true*s appears on the inputs.

The **categorical problems** come from Spector *et al.*’s work on evolving algebraic terms [11] and dwell in the ternary domain: the admissible values of program inputs and outputs are $\{0, 1, 2\}$. The peculiarity of these problems consists in using only one binary instruction in the programming language; for the a_1 algebra, the semantics of that instruction is defined in Table 3c. We refer the reader to [11] for the definitions of the remaining algebra problems.

For each of the five algebras considered here, we consider two tasks. In *discriminator term* tasks (D -* in Table 2), the goal is to synthesize an expression (using only the one given instruction) that accepts three inputs x, y, z and realizes the function given in Table 3a. Given three inputs and ternary domain, this gives rise to $3^3 = 27$ fitness cases for these benchmarks.

The second task defined for each of the algebras (M -* in Table 2), consists in evolving a so-called *Mal’cev term*, i.e., a ternary term that is equivalent to the one given in Table 3b. This condition specifies the desired program behavior only

Table 3. The algebra problems: (a) discriminator problem, (b) Mal’cev problem, (c) exemplary algebra (named a_1 in [11])

$$(a) \ t(x, y, z) = \begin{cases} x & \text{if } x \neq y \\ z & \text{if } x = y \end{cases} \quad (b) \ m(x, x, y) = m(y, x, x) = y \quad (c) \ \begin{array}{c|ccc} a_1 & 0 & 1 & 2 \\ \hline 0 & 2 & 1 & 2 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 \end{array}$$

for the indicated combinations of inputs, and the desired value for all distinct inputs is not determined. As a result, there are only 15 fitness cases in our Mal’cev tasks, the lowest number of all considered benchmarks.

The **regression problems** considered here come from [7] and include both univariate and multivariate target functions. The univariate ones (*Keij1*, *Keij4*, *Nguy3..7* and *Sext*) use 20 tests uniformly distributed in the $[-1, 1]$ interval, except for the *Keij4* benchmark which uses the $[0, 10]$ interval. The remaining problems are predominantly bivariate, and involve $5 \times 5 = 25$ fitness cases uniformly distributed on the two-dimensional grid. The only exception is *Keij5*, which hosts three input variables, with $4 \times 4 \times 4 = 64$ fitness cases distributed equidistantly in the cube. For other details on these benchmarks, see [7].

Results for Limited Number of Evaluations. In this experiment, the computational budget allocated to every run is 250,000 evaluations, i.e., a population of 1,000 programs evolves for 250 generations. Because of the large number of benchmarks and limited space, we discuss only the aggregated outcomes. To this aim, we rank the three considered configurations according to the success rate on every benchmark independently. Next, we average the ranks across benchmarks.

When averaged over **all benchmarks**, the resulting ranks are FI: 1.6, F: 2.19, and FxI: 2.21 (the lower rank, the better). To assess the statistical significance of this outcome, we used the Friedman’s test for multiple achievements of multiple subjects which, compared to ANOVA, does not require the distributions of variables in question to be normal. The p -value of 0.00124 strongly indicated that at least one method performed significantly different from the remaining ones. A post-hoc analysis using symmetry test [2] determined that the difference between F and FxI is statistically insignificant, but FI significantly outranks them both ($p = 0.03$).

We can conclude thus that augmenting the conventional training signal (program error) with our information-based behavioral measure that depends on internals of program execution (information consistency) brings substantial benefits to a GP search algorithm. However, this seems to hold only when the behavioral information is provided as a separate search objective (FI setup). Aggregation of conventional fitness with information consistency (FxI setup), at least in the specific multiplicative manner used here (Eq. 2) does not make the search more efficient. Possible explanation is that F , defined as city-block distance for regression, and normalized Hamming distance for the remaining two domains, may assume radically different ranges on different problems. As a result, the trade-off between F and I varies across domains and can be difficult to control.

FI fared the best on the **categorical problems**, where it came on top on nine out of ten benchmarks, so that its average rank was 1.2 there. Remarkably, this is also the only domain in which FxI ranked on average better than F (2.05 vs. 2.75). This may suggest that the trade-off between F and I was just right for these problems. For **regression problems**, the differences between all three methods were the least prominent (FI: 1.82, F: 2.0, FxI: 2.18), which was expected, as discrete entropy cannot capture similarities between values for these continuous problems.

On the **Boolean domain**, the behavioral methods performed relatively bad (FI: 1.62, F: 1.88, FxI: 2.5), which seems surprising given the discrete nature of these problems. We come up with three mutually nonexclusive hypotheses to account for this. Firstly, note that the Boolean instruction set does not feature negation (Table 2). Let us consider a program p that evolves a subexpression p' which produces exactly the negated value of desired output Y . In terms of I , p' is perfectly consistent with Y , so $I(p) = 0$. However, without negation in the instruction set, it may be difficult to extend p' with a suffix that would turn it into Y . A way to achieve this is p' *nor false* or p' *nand true*, but those Boolean constants require a separate subprogram to synthesize them.

The other hypothesis starts with the observation that Boolean problems feature the largest number of input variables in our benchmark suite ($v \geq 6$, while $v \leq 3$ for the other domains, Table 2). As no input variable is redundant in these problems, evolution has to produce a subprogram comprising at least v tree leaves (terminals), and thus $2v - 1$ nodes, to possibly bring I to zero. In general, to obtain competitive values of I , relatively large subtrees featuring most input variables have to be synthesized.

Last but not least, the random variables that I is based on, by being binary for the Boolean problems, are least discriminating in terms of entropy. As an example, a binary random variable observed for five training examples can have only one of three possible values of entropy, while an analogous number for a ternary variable (like those used in our categorical problems) is five. As a result, I can be more fine-grained for domains that feature multi-valued variables, even if the actual number of examples is low (like in our categorical problems).

Results for Limited Time Budget. To take into account the overhead of calculating consistency measure, in this experiment the computational budget allocated to every run was 600 seconds. For this setup, the average ranks w.r.t. success rate are: FI: 1.77, F: 2.06, FxI: 2.17. Though FI leads again, this time Friedman test is inconclusive at 0.05 level. However, its relatively low p -value (0.08) suggests that significance could be attained given a larger suite of benchmarks.

4 Summary

We proposed a measure for characterizing the internal program behavior in terms of its consistency with the desired output, which can conveniently be used to promote certain program behaviors without specifying them explicitly. The algorithms that involve this measure can be thus said to implicitly perform *problem*

decomposition, which normally requires an expert who explicitly splits a task into subtasks. Together with methods reported elsewhere [5], we consider information consistency as promising way towards scalable program synthesis.

Acknowledgment. K. Krawiec acknowledges support from the NCN grant no. DEC-2011/01/B/ST6/07318, and A. Solar-Lezama from grant no. NSF-CCF-1161775.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
2. Hollander, M., Wolfe, D.: *Nonparametric Statistical Methods*. A Wiley-Interscience Publication. Wiley (1999)
3. Jiang, J.-H.R., Lee, C.-C., Mishchenko, A., Huang, C.-Y.R.: To sat or not to sat: Scalable exploration of functional dependency. *IEEE Transactions on Computers* 59(4), 457–467 (2010)
4. Knowles, J.D., Watson, R.A., Corne, D.W.: Reducing local optima in single-objective problems by multi-objectivization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) *EMO 2001*. LNCS, vol. 1993, pp. 269–283. Springer, Heidelberg (2001)
5. Krawiec, K., Swan, J.: Pattern-guided genetic programming. In: Blum, C., et al. (eds.) *GECCO 2013: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, Amsterdam, The Netherlands, July 6–10, pp. 949–956. ACM (2013)
6. Luke, S.: *ECJ evolutionary computation system* (2002), <http://cs.gmu.edu/ecclab/projects/ecj/>
7. McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., O’Reilly, U.-M.: Genetic programming needs better benchmarks. In: Soule, T., et al. (eds.) *GECCO 2012: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference*, Pennsylvania, USA, July 7–11, pp. 791–798. ACM (2012)
8. McPhee, N.F., Ohs, B., Hutchison, T.: Semantic building blocks in genetic programming. In: O’Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) *EuroGP 2008*. LNCS, vol. 4971, pp. 134–145. Springer, Heidelberg (2008)
9. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I*. LNCS, vol. 7491, pp. 21–31. Springer, Heidelberg (2012)
10. Singh, R., Singh, R., Xu, Z., Krosnick, R., Solar-Lezama, A.: Modular synthesis of sketches using models. In: McMillan, K.L., Rival, X. (eds.) *VMCAI 2014*. LNCS, vol. 8318, pp. 395–414. Springer, Heidelberg (2014)
11. Spector, L., Clark, D.M., Lindsay, I., Barr, B., Klein, J.: Genetic programming for finite algebras. In: Keijzer, M., et al. (eds.) *GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, Atlanta, GA, USA, July 12–16, pp. 1291–1298. ACM (2008)

On Effective and Inexpensive Local Search Techniques in Genetic Programming Regression

Fergal Lane, R. Muhammad Atif Azad, and Conor Ryan

CSIS Department, University of Limerick, Ireland
{Fergal.Lane,Atif.Azad,Conor.Ryan}@ul.ie

Abstract. Local search methods can harmoniously work with global search methods such as Evolutionary Algorithms (EAs); however, particularly in Genetic Programming (GP), concerns remain about the additional cost of local search (LS). One successful such system is *Chameleon*, which tunes internal GP nodes and addresses cost concerns by employing a number of strategies to make its LS both effective and inexpensive. Expense is reduced by an innovative approach to parsimony pressure whereby smaller trees are *rewarded* with local search opportunities more often than bigger trees.

This paper investigates three new extensions to Chameleon’s original simple setup, seeking ways for an even more effective local search. These are: trying alternative, more cost-reflective parsimony measures such as *visitation length* instead of tree size; varying the reward function to more gently incentivize parsimony than that in the original setup; and having more tuning earlier in runs when smaller trees can be tuned more cheaply and effectively. These strategies were tested on a varied suite of 16 difficult artificial and real-world regression problems. All of these techniques improved performance.

We show that these strategies successfully combined to cumulatively improve average test RMSE by 31% over the original and already effective Chameleon tuning scheme. A minimum of 64 simulations were run on every problem/tuning setup combination.

1 Introduction

GP regression/classification is potentially a very attractive candidate for combining with local search (LS) strategies. This is because most of the expense of fitness evaluations for points in the locality of already evaluated points can be avoided. Caching strategies, where intermediate node evaluations are stored in a local cache, make this possible. That LS approaches are seldom used with GP is, therefore, somewhat surprising, but is perhaps explained by concerns regarding excessive computational overheads or worries about complex implementational details. While several researchers have previously carried out work in this area, the Chameleon lifetime learning system [2,3] is, nonetheless, a relatively rare example of a GP augmented with LS. Chameleon further distinguishes itself with a low-cost and intuitive LS approach. Its local search mechanism tunes

internal GP tree nodes. Moreover, rather than explicitly penalizing bigger trees in a stick-like manner, Chameleon allocates node tuning effort in a carrot-like fashion by rewarding smaller trees with greater tuning opportunities using a simple linear reward function. This successfully encourages and maintains more parsimonious trees. The Chameleon system has significantly outperformed state-of-the-art GPs [2,3]. This was all achieved using a very simple, cheap, and easy to implement LS node tuning mechanism.

This paper sets out a number of new ways in which Chameleon’s approach to node tuning can be even further improved. In our investigations, we were always very much concerned with controlling computational costs. Our goal was to better learn how this tuning effort might be even more effectively directed. We investigated alternative parsimony measures instead of tree size that better reflect tree tuning costs. We varied the gentleness/severity of how tuning effort was allocated according to parsimony. *When* tuning is performed can have a big impact on cost. We, therefore, investigated reallocating more tuning to earlier in runs where smaller trees can be more cheaply and effectively tuned.

The paper is laid out as follows. Section 2 gives a brief survey of the previous uses of LS in GP and, in particular, gives a basic grounding in the Chameleon lifetime learning system. Section 3 details the experimental setup used to test our new tuning strategies. In section 4 we describe these tuning schemes in greater detail and give experimental results and analysis for the individual strategies. Section 5 describes how these individual methods harmoniously combine to give even greater cumulative performance gains. We end with a final section laying out our conclusions and describing some potential future research directions.

2 Related Work

2.1 Local Search in Genetic Programming

While the use of LS with GP has been relatively rare, the idea of augmenting the global search capabilities of stochastic population optimizers with LS, as typified by the Memetic Algorithm (MA) approach, has been a far more frequent theme in the EA field (see [14] for an overview of MAs and how LS has been used in conjunction with EAs). Radcliffe [17] has previously argued that the MA approach is most attractive when fitness functions are “decomposable”, i.e. when points in the locality of an already visited individual can be evaluated relatively more cheaply than for arbitrary locations (the TSP is a good example where this “decomposability” holds and has been exploited by a state-of-the-art solver [1]). This same pattern can be made to hold for GP regression and classification merely by caching intermediate calculations when evaluating trees. When a part of a tree is changed, the cached calculations for the unaltered regions of the tree do not need to be recomputed, and can simply be reused.

The rather infrequent use of LS in GP is, therefore, a little puzzling. Nonetheless, there has been some work in this area. Space considerations preclude giving an extensive list (see [3] for a comprehensive review). Even when LS is used, it is unusual to see caching being used to reduce costs ([11] [18] are rare exceptions).

[10] was also concerned with controlling costs, limiting application of LS to the best population member for just one hillclimbing (HC) step. Most work, though, has not considered cost. Some research [7] has used GP to directly (co-)evolve the actual LS heuristics used. Other work has focused on tuning external GP nodes; [9] gives a detailed survey of past work on tuning GP numeric constants.

Other research, which more closely relates to Chameleon, has focused on tuning internal GP nodes. Often these studies have used mutation or crossover in a HC fashion. Majeed [11] studied a form of HC “context aware” crossover, where the incoming subtree was always inserted at the best possible site in the recipient. [15] formulated a minimal-change mutation operator and tested it with SA/HC approaches (later successfully using this HC approach with GP crossover [16]). [4] combined crossover with a form of elitist brood selection. Zhang [21] again used crossover in a broadly similar elitist HC fashion, but also incorporated an interesting “looseness” mechanism, which helped discover and protect good GP building blocks from the disruptive effects of crossover. Whereas these approaches employ mutation or crossover in a LS fashion, Chameleon uses a wholly separate node-tuning phase, which the upcoming subsection will now describe.

2.2 Chameleon

The Chameleon lifetime learning system tunes *internal* GP nodes in a Lamarckian fashion. When Chameleon tunes an internal node, n , it simply tries out one-by-one all valid function set possibilities for that position and then keeps the best one. When a GP node is altered, there is no need to completely recalculate the entire fitness evaluation. Chameleon uses caching techniques to store all intermediate node evaluations, so it merely suffices to update the $\text{depth}(n)$ cache locations for the nodes in the direct line from there to the top of the tree. Therefore, for a node, its tune cost is proportional to its depth. This caching principle is one of the two foundations underpinning the effectiveness of the Chameleon system. Indeed, earlier papers [2,3] found that even node tuning by itself leads to big performance gains. An *exhaustive tuning* scheme where every node in the population is tuned exactly once significantly outperformed standard GP.

The second major foundation underlying Chameleon is an innovative incentive-based approach to controlling bloat. The related questions of overfitting, generalization and bloat are important issues for any regression/classification paradigm to address. Rather than having a stick-like penalty function, Chameleon instead uses its node tuning mechanism in a carrot-like fashion to promote parsimony. Under this indirect approach, no large individual is explicitly punished; instead smaller individuals are rewarded with greater node tuning opportunities (results have shown that this also improves generalization and reduces overfitting).

2.3 Chameleon’s Parsimony Reward Scheme

Chameleon uses a very straightforward tuning reward scheme, called *probabilistic tuning*, to encourage more parsimonious trees. Individual GP trees are allocated with node tunes according to a simple linear reward function (truncated to keep

Table 1. Reward Function Formulae

Original Probabilistic Tuning
$Trunc_{[0,1]} [1.5 - size(t)/\overline{size}]$
Probabilistic Tuning (Variant with Slope = -0.5)
$Trunc_{[0,1]} [1 - 0.5 size(t)/\overline{size}]$

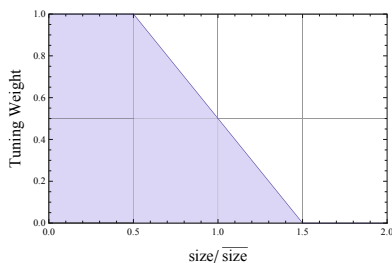


Fig. 1. Original Probabilistic Tuning Reward Function

its output within the probabilistic $[0, 1]$ range) given in the first row of table 1. An average-sized tree receives a weight of 0.5 (so each of its internal nodes has a 50% chance of being tuned). This probability will be higher for smaller trees and lower for larger trees. Smaller individuals can receive up to double the tuning effort of average-sized population members. Larger individuals get less tuning, eventually dwindling to zero when they exceed 150% of average size (see Fig. 1). The probability of any node being tuned depends on the weight given to the individual tree by the tuning reward function. This function assigns a probability to each individual in the population.

In contrast, the *exhaustive tuning* scheme simply tunes nodes in a blanket fashion (tuning every node in the population exactly once). The marriage in probabilistic tuning between efficient node tuning and an incentivized parsimony-reward mechanism, which more carefully distributes tuning effort, generally leads to even better performances than just indiscriminate exhaustive tuning alone [2,3] (a pattern which is again confirmed by our experimental results).

3 Experimental Setup

This section gives a detailed description of the experimental setups used to test our new tuning strategies and comparison base cases. Table 2 supplies details of the suite of 16 problems (11 artificial and 5 real-world) used in our experiments.

GP Parameter Settings. The GP parameter settings used in our experiments are given in table 3. Almost identical settings were used to compare the various tuning setups and baseline standard GP. We endeavoured to ensure a scrupulously fair like-for-like comparison in terms of computational resources. This motivated our use of a *nodes processed* statistic as the termination criterion. This was a running count roughly equivalent to the computational resources used so far in the simulation. $size(t)$ was added to the statistic after every evaluation (where each node is visited once) of a tree t . Each time the tuning process investigated a new function set possibility for a node n , the count was increased by $depth(n)$ (corresponding to the number of node recalculations needed).

In our tuning scheme experiments, when the *nodes processed* count reaches 1.6 million the run is terminated. The *nodes processed* termination mechanism

Table 2. Problem Set

Problem(s)	# Vars	Sampling Range(s)	Total Dataset Size	Training Subset Size	Further Details
Pagel	2	U[-5,5]	3000	1000	See McDermott [12], which surveys commonly used GP problems, for the various original proposers, sources and references for these
Korns1	5	U[-50,50]	3000	1000	
Keijzer 11, 12, 13, 14, 15	2	U[-3,3]	3000	1000	
Vladislavleva4	5	U[-0.25, 6.35]	3000	1000	
Vladislavleva5	3	x,z: U[-0.05,2.1] y: U[0.95,2.05]	3000	1000	
Vladislavleva7	2	U[-0.25, 6.35]	3000	1000	
Vladislavleva8	2	U[0.05, 6.05]	3000	1000	
Dow Chemical	57	Supplied Dataset	1066	747	
Concrete Compressive Strength	8	Supplied Dataset (UCI)	1030	350	UCI Machine
Housing	13		506	200	Learning Repository
Parkinsons Telemonitoring(Total and Motor)	20		5875	1500	[6]

Table 3. Parameter Settings

Population Size	500 (tuning), 1500 (no tuning)	Tree Initialization	Ramped-Half-and-Half (max. init. depth = 4)
Replacement	Steady State, Inverse Tournament	Termination	When 1.6 million nodes processed
Mutation Rate Per Node	0.05	Ephemeral Random Constants (ERCs)	Special terminal nodes in GP used to hold numeric constants. $ ERC = 50$, generated uniformly randomly in [-5,5]
Crossover Rate	0.9		
Crossover	Subtree Crossover		
Tournament Size	2	Number of Simulations	Always at least 64 individual runs per problem for every tuning setup (≥ 128 for baseline and important tuning setups)
Selection	Tournament Selection		
Functions Used	{+, -, ×, ÷, pow, exp, log, cos, sin}		
Terminal Set	{Input Variables} \cup ERCs		

largely ensured similar runtimes on individual problems. However, it’s still an approximate rather than perfect gauge of computational cost. Due to general overheads, tuning runs were still taking on average 25% longer than for standard GP, so we boosted the resources for standard GP runs. We allowed standard GP 25% more runs; for every 4 tuning runs we ran 5 standard GP runs and then simply discarded the standard GP run with the worst performance.

GP runs using tuning actually perform *fewer* absolute number of evaluations than standard non-tuning GP without caching. Therefore, it seemed appropriate to have different population size settings for the tuning and the non-tuning cases. Standard GP worked best with a population size of 1500. A population of 500 was found to be a good general setting for the tuning cases. Otherwise, all other settings were identical.

Geometric Mean as a Summary Statistic. Lack of space prevented us from including detailed RMSE breakdowns for all problems. This motivated us to use the *geometric mean* to summarize *test* RMSE performance over the entire test suite. The geometric mean of a set $\{a_1, \dots, a_n\}$ is given by $(\prod_{i=1}^n a_i)^{1/n}$, which is also equivalent to $\exp(\sum_{i=1}^n \ln(a_i + b)) - b$ when $b = 0$.¹ The geometric mean was chosen as it is resistant to outliers, is particularly meaningful when one is

¹ We actually used a very slightly modified form with $b = 0.0001$ to successfully cope with zero RMSE values (this is a long-established strategy, e.g. see [20]).

working with ratios, and still gives meaningful averages even when combining groups with widely varying scales and ranges [5].

4 Node Tuning Strategies

We investigated three natural extensions to Chameleon’s original probabilistic tuning scheme. Cost considerations were an important motivation (influenced by Radcliffe’s [17] argument that MA approaches are most fruitful when local moves can be made relatively cheaply).

4.1 Cost-Oriented Parsimony Measures

Because balanced trees are cheaper to tune [3], we wanted to test measures more reflective of the actual costs of processing GP trees and that factored in tree skewness. A prior example from GP would be *visitation length* [8][19]. Visitation length equals size plus *path length* of a GP tree. *Path length* is the sum of the depths of all nodes in a tree. Both path length and visitation length increase with increasing tree size, but also capture tree skewness. Highly imbalanced trees with identical sizes will have higher values for these measures. Table 4 lists the parsimony measures studied and also gives their overall mean test RMSE performances over the test suite (here $N(t)$ and $I(t)$ are the sets of all nodes and all internal nodes, respectively, of a tree t). Since as of yet, Chameleon only tunes internal nodes, we tested some internal node versions of these measures, e.g. *internal path length* (IPL) is the sum of depths of all internal nodes in a tree.

Table 4. Parsimony Measures and Performances

Parsimony Measure	Formula	Mean Test RMSE	
		Original	Modified Slope
Standard GP	-	0.955±0.022	
None (Exhaustive Tuning)	-	0.574±0.012	
Size (Standard Prob. Tuning)	$ t $	0.456 ± 0.024	0.441±0.036
Number of Internal Nodes	$ I(t) $	0.460±0.021	0.429±0.029
Path Length	$\sum_{n \in N(t)} \text{depth}(n)$	0.423±0.019	0.403±0.024
Internal Path Length	$\sum_{n \in I(t)} \text{depth}(n)$	0.409±0.017	0.374±0.016
Visitation Length	$ t + \sum_{n \in N(t)} \text{depth}(n)$	0.409±0.017	0.400±0.025
Internal Visitation Length	$ t + \sum_{n \in I(t)} \text{depth}(n)$	0.427±0.018	0.380±0.024

It is clear from these results that performance gains are possible by using parsimony measures more directly related to tuning costs. IPL, highlighted in the table, gave the best *test* RMSE performance (and would most closely mirror tree tuning costs). IPL gave a 10.3% performance gain over standard probabilistic tuning.

Varying the Slope of the Reward Function. The original linear *probabilistic tuning* reward function with its slope of $a = -1$ gave good performances. However, we were interested in seeing how performance might vary if we changed the slope of this line and the gentleness/severity of the parsimony pressure. Although the available space prohibits us from reporting detailed results here, we did find that somewhat more gently incentivizing parsimony gave superior performances to the original form or no incentivization at all. An intermediate slope value of $a = -0.5$ consistently gave best performances for several different parsimony measures out of a wide range of a values tested (with corresponding formula for $a = -0.5$ given in the final row of table 1).

The final column of table 4 gives performances for the various parsimony measures under this tuning scheme. IPL was still the best performer (its gain over probabilistic tuning significant at a 99% level using the two-tailed Student's t-test). However, under both reward schemes, there was not a statistically significant margin separating some of the better performers. Hence, more investigation will be needed to see if we can clearly separate out one of these measures as having a consistent advantage over the others.

4.2 More Tuning Earlier in Runs

Generally, tree sizes increase as GP runs progress. For both probabilistic and exhaustive tuning, population tune costs rise correspondingly. Therefore, most of the available tuning effort tends to be expended tuning larger deeper trees in later generations (until our fixed *nodes processed* budget runs out). Some preliminary investigations hinted that tuning effort might be more effective when biased towards the start of a run ([13] also indicated that most useful work at the top of trees is done early in runs). This provided some motivation to look at strategies that shift more tuning resources to earlier generations. We include here one simple and very effective tuning strategy that redistributes tuning effort at the generational level. This *constant nodes* strategy imposes the requirement that the average number of nodes tuned in every generation remains constant. In earlier generations with smaller trees, the average tuning per tree is then likely to be far higher.

Our technique normalizes the sum of all population tree tuning weights so that a constant average number of nodes is always tuned per generation. A single parameter n determines the fixed generational tuning budget. n corresponds to the average number of nodes tuned per tree per generation. Hence, in a population with p trees, an average of $n \times p$ nodes will always be tuned (particular individuals may not get an exact budget of n tunes due to unequal prior weights, this is only an overall average for individuals in the population). An added complication is that tree weights may now exceed 1.0 (we deal with this by allowing multiple cycles in a tree tuning until its weight reaches zero).

As can be seen from Fig. 2, an n value of around 25 for the tuning budget works best, leading to a substantial 19.3% performance boost over the equivalent probabilistic tuning setup (significant at a 95% level using the two-tailed Student's t-test).

Table 5. Baseline and Combined Strategy Performances

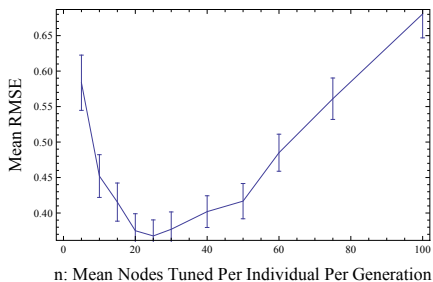


Fig. 2. Performances for the *Constant Nodes* Strategy with Different n Values

Tuning Setup			Overall RMSE
Standard GP			0.955±0.022
Exhaustive Tuning			0.574±0.012
Probabilistic Tuning			0.456±0.024
SLOPE	IPL	CN	Overall RMSE
y	n	n	0.441±0.036
n	y	n	0.409±0.017
n	n	y	0.368±0.023
y	y	n	0.374±0.016
y	y	y	0.315±0.017

5 Combining Tuning Strategies

These individual tuning strategy categories all gave moderate performance gains when used in isolation. However, they combined harmoniously and relatively seamlessly to produce a far bigger cumulative performance boost. The best parameter settings for these strategy types seemed to remain broadly similar whether used in isolation or in concert with others.

A combination of all of the successful strategy types eventually gave a 31% test RMSE improvement over probabilistic tuning and a 67% improvement over the standard GP base case (both significant at a 99.9% level). Table 5 gives baseline performance comparisons and then shows the performance gains for various combinations of the new strategy types. The combination of all three strategies (final row) was easily the best performer. Changing the parsimony measure from size to IPL resulted in a 10.3% performance gain. Varying the reward function slope to $a = -0.5$ (SLOPE) pushed the combined performance improvement to 18%. Finally, adding in the *constant nodes* (CN) strategy with $n = 25$ resulted in a cumulative 31% improvement over the original probabilistic tuning scheme (significant at a 99.9% level). Somewhat surprisingly, the performance gains were almost additive. The performance boost for the final combination was 94% of the sum of individual gains for the three basic strategies.

Some individual problem RMSE breakdowns are given in table 6. Probabilistic tuning was superior to standard GP on all 16 problems (this ordering was 99% significant for 12 problems, 95% significant for 2, and not significant for 2). However, the final combined strategy was, in turn, superior to probabilistic tuning on every problem in the test suite (this ordering was 99% significant for 12 problems, 95% significant for 1, and not significant for 3). The superiority of the final combined strategy to standard GP was 99% significant for all problems.

Table 6. Mean Test RMSE Breakdowns on Individual Problems

Tuning Scheme	Pagie		Korns		Keijzer					Vladislavleva				DowChem	Concrete	Housing	Parkinsons	
	1	1	11	12	13	14	15	4	5	7	8	Total	Motor					
Standard GP	0.237	0.316	0.552	2.268	0.095	0.144	0.772	0.551	0.384	2.295	0.712	0.327	11.711	5.286	9.629	7.260		
Prob. Tuning	0.167	0.009	0.280	0.954	0.001	0.119	0.764	0.541	0.218	1.662	0.621	0.313	10.516	5.013	8.423	6.444		
All Combined	0.116	0.003	0.163	0.423	0.000	0.073	0.699	0.512	0.132	1.244	0.455	0.292	9.650	4.875	7.846	6.059		

6 Conclusions

This paper has tread some new ground in terms of node tuning strategies. Results garnered on this problem suite indicate that significant performance gains are possible by more judiciously expending tuning effort.

Using internal path length as a parsimony measure boosted *test* RMSE performance by 10.3%. Moderating parsimony pressure (changing the reward function slope to -0.5) increased performance by another 7.7%.

However, the greatest leap in performance was produced from the *constant nodes* strategy. Adding this strategy boosted performance by a further 13% to give a cumulative improvement for all strategies of 31%.

We have again confirmed on a new problem test set the benefits of Chameleon’s node tuning approach. Even after carefully equalizing computational costs, the exhaustive and particularly probabilistic tuning schemes continue to show substantial performance advantages over standard GP. Our new strategies combine to produce significant improvements over even these more established schemes. Of course, it will be necessary to see whether such strategies continue to be useful in other regression (and also classification) environments.

6.1 Future Research Directions

The *constant nodes* scheme (concerned with *when* to tune) gave individually the greatest performance gain. Tuning strategies of this nature certainly merit further investigation. We are also currently getting some promising preliminary results from strategies that internally redistribute tuning effort *within* trees (bi-asing tuning towards shallower cheaper-to-tune nodes).

Chameleon tuning of external nodes (ERCs or input variables) has not yet been properly investigated. Some means of narrowing down the range of variable substitution choices for a potentially large input variable set, perhaps by exploiting problem structure or variable dependencies, would likely be necessary.

We also plan to investigate whether relatively coarse-grained internal node tuning and finer-grained gradient-based constant tuning can work together in a complementary fashion.

References

1. Applegate, D., Cook, W., Rohe, A.: Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing* 15(1), 82–92 (2003)
2. Azad, R.M.A., Ryan, C.: Abstract functions and lifetime learning in genetic programming for symbolic regression. In: *Genetic and Evolutionary Computation Conference (GECCO-2010)*, pp. 893–900. ACM (2010)

3. Atif, R.M., Azad, C.R.: A simple approach to lifetime learning in genetic programming based symbolic regression. *Evolutionary Computation* 22(2), 287–317 (2014)
4. Bhardwaj, A., Tiwari, A.: A Novel Genetic Programming Based Classifier Design Using a New Constructive Crossover Operator with a Local Search Technique. In: Huang, D.-S., Bevilacqua, V., Figueroa, J.C., Premaratne, P. (eds.) ICIC 2013. LNCS, vol. 7995, pp. 86–95. Springer, Heidelberg (2013)
5. Clark-Carter, D.: Geometric mean. In: *Encyclopedia of Statistics in Behavioral Science*. Wiley (2005)
6. Frank, A., Asuncion, A., et al.: UCI machine learning repository (2010)
7. Fukunaga, A.S.: Evolving local search heuristics for SAT using genetic programming. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 483–494. Springer, Heidelberg (2004)
8. Keijzer, M., Foster, J.: Crossover bias in genetic programming. In: Ebner, M., O’Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 33–44. Springer, Heidelberg (2007)
9. Kommenda, M., Kronberger, G., et al.: Effects of constant optimization by nonlinear least squares minimization in symbolic regression. In: *Genetic and Evolutionary Computation Conference (GECCO 2013)*, pp. 1121–1128. ACM (2013)
10. Krawiec, K.: Genetic programming with local improvement for visual learning from examples. In: Skarbek, W. (ed.) CAIP 2001. LNCS, vol. 2124, pp. 209–216. Springer, Heidelberg (2001)
11. Majeed, H., Ryan, C.: On the constructiveness of context-aware crossover. In: *Genetic and Evolutionary Computation Conference (GECCO-2007)*, pp. 1659–1666. ACM (2007)
12. McDermott, J., White, D.R., et al.: Genetic programming needs better benchmarks. In: *Genetic and Evolutionary Computation Conference (GECCO 2012)*, pp. 791–798. ACM (2012)
13. McPhee, N.F., Hopper, N.J.: Analysis of genetic diversity through population history. In: *Genetic and Evolutionary Computation Conference (GECCO 1999)*, vol. 2, pp. 1112–1120 (1999)
14. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms. In: *Handbook of metaheuristics*, pp. 105–144. Springer (2003)
15. O’Reilly, U.-M., Oppacher, F.: Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 397–406. Springer, Heidelberg (1994)
16. O’Reilly, U.-M., Oppacher, F.: A comparative analysis of genetic programming. In: *Advances in Genetic Programming 2*, ch. 2, pp. 23–44. MIT Press (1996)
17. Radcliffe, N.J., Surry, P.D.: Formal memetic algorithms. In: Fogarty, T.C. (ed.) AISB-WS 1994. LNCS, vol. 865, pp. 1–16. Springer, Heidelberg (1994)
18. Scoble, A., Johnston, M., Zhang, M.: Local search in parallel linear genetic programming for multiclass classification. In: Thielscher, M., Zhang, D. (eds.) AI 2012. LNCS, vol. 7691, pp. 373–384. Springer, Heidelberg (2012)
19. Smits, G.F., Kotanchek, M.: Pareto-front exploitation in symbolic regression. In: *Genetic Programming Theory and Practice II*, pp. 283–299. Springer (2005)
20. Williams, C.B.: The use of logarithms in the interpretation of certain entomological problems. *Annals of Applied Biology* 24(2), 404–414 (1937)
21. Zhang, M., Gao, X., Lou, W.: A new crossover operator in genetic programming for object classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 37(5), 1332–1343 (2007)

Combining Semantically-Effective and Geometric Crossover Operators for Genetic Programming

Tomasz P. Pawlak

Institute of Computing Science, Poznan University of Technology, Poznań, Poland
tpawlak@cs.put.poznan.pl

Abstract. We propose a way to combine two distinct general patterns for designing semantic crossover operators for genetic programming: geometric semantic approach and semantically-effective approach. In the experimental part we show the synergistic effects of combining these two approaches, which we explain by a major fraction of crossover acts performed by geometric semantic crossover operators being semantically ineffective. The results of the combined approach show significant improvement of performance and high resistance to a premature convergence.

Keywords: Semantics, taxonomy, neutrality, brood selection, experiment.

1 Introduction

Genetic Programming (GP), as a method of automatic induction of discrete structures from an arbitrary given set of building blocks, guided by a fitness function has been known for over 25 years [13,29]. Among other applications, like synthesis of electronic circuits [14,15] or design of 3D structures [21,1], GP is mostly used for induction of programs from a set of instructions. To accurately assess the program's fitness, it must be run multiple times on a set of program inputs to compare the produced outputs with the desired target outputs. Typically the divergence between the target output and the actual program output is measured by a minimized fitness function and the pair of program input and the target program output is referred to as *fitness case* [30,19,25,16,8].

The tuple that consists of a program output for each fitness case is known as *semantics*¹. Consequently, the target program output extracted from a set of fitness cases is also a semantics, however to distinguish this one, we refer to it as *target semantics*. The main role of program semantics is to describe program behavior in a concise way.

In recent years we observe a growing interest in semantics in genetic programming. Most applications of semantics are related to designing novel genetic operators, such as crossover [30,19,25,16,27,2] and mutation [25,27,26,4], however

¹ The common definition involves a vector instead of tuple [30,19,25,16,8], however vector is by definition a tuple of numbers and we consider programs that operate on an arbitrary data, thus we keep the tuple in the definition as more general.

some researchers also analyze the impact of semantic initialization of the initial population [3,10] and selection [8].

In this paper we focus on semantic crossover operators. We distinguish two general patterns of designing semantic crossover operators, which are *semantically-effective* and *geometric* patterns. In the first one semantics is used to reject the offspring having the same or similar semantics to any of its parents, while in the second one the crossover is guided on producing offspring having semantics that is a certain combination of semantics of parents. We present the way to combine these two approaches and experimentally confirm advantages of this combination.

2 Taxonomy of Semantic Crossover Operators

We divide the collection of semantic crossover operators for tree-based GP, depending on they use program semantics for either providing *effective* changes, or exploiting *geometric* properties of search space. This taxonomy shall not be confused with the one proposed in [32], since the latter one divides semantic operators into operators making syntactic changes with *indirect* impact on the semantics, and manipulating syntax to *directly* influence semantics.

Naturally we do not include in this taxonomy non-semantic crossover operators, such as tree-swapping crossover [13], context-preserving crossover [6], one-point crossover [28], size-fair crossover [20] and depth-based crossover [9].

Semantically-Effective Crossover Operators

The first class of crossover operators consists of operators that use semantics mainly for the purpose of rejecting offspring that are semantically equal to any of their parents. In this sense these operators produce *semantically-effective* offspring w.r.t. its parents by rejecting the *ineffective* ones. They are commonly referred in the literature to as ‘semantically-driven’ [2] or ‘semantic-aware’ [30], since to the best of our knowledge they were historically the first semantics-based crossover operators. However the historical names are too general, thus we decided to refer to these operators to as *semantically-effective*.

The term ‘semantically-effective’ shall not be considered as an opposition to *neutrality* in GP, which refers to two notions: (i) neutral genes, e.g., introns [12,7], and (ii) mutation performing genotypic changes that are not reflected in phenotype [12] or fitness change [7]. In contrast semantically effective crossover (i) does not negate that an offspring may contain introns, and (ii) does not perform a (mutational) change in the genotype, instead it combines genotypes of two parents. Moreover the ‘effectiveness’ of crossover is defined on the semantic level, which is fundamentally different from what is meant by ‘phenotype’ in these early studies [12,7], i.e., a syntax tree.

One of the first of semantically-effective crossovers is *semantically driven crossover* (SDC) by Beadle et al. [2], which involves reduced ordered binary decision diagrams (ROBDD) as a representation for semantics of Boolean programs.

Thanks to the uniqueness property of ROBDD, their operator is able to prevent of crossing over parents having semantically equal subtrees rooted at crossover points. Later Quang et al. [30] extended this idea to the domain of real function synthesis by proposing *semantics aware crossover* (SAC). Their operator discards all crossover acts of parents, whose subtrees rooted at crossover points are semantically less distant than a given threshold, called semantic sensitivity. Thus SDC and SAC are conceptually equivalent, however adapted for different domains.

To some extent another operator by Quang et al. [30], namely *semantic similarity based crossover* (SSC) can be also included in this class of operators. SSC operates similarly to SAC, however it adds second threshold to the semantic distance between parents' subtrees to prevent breeding offspring unrelated to its parents. In this sense SSC reduces the chaotic characteristics of crossover. The same authors [31], arguing that not only the effectiveness but also the magnitude of change is important, proposed the *most semantic similarity based crossover* (MSSC), which operate similarly to SSC, however instead of using the second threshold, it promotes the exchange of subtrees that are the most similar but different.

Semantic Geometric Crossover Operators

We distinguish a separate class of crossover operators that guide the crossover act to produce offspring having semantics that is a kind of geometric combination of semantics of its parents. Such a combination is usually a point on the segment spanned over semantics of parents. If crossover operator guarantees that the semantics of offspring lies on this segment, we refer to it as *geometric crossover*. This definition is consistent with [24,18,27]. Although most of the operators presented in this section only approximate geometric crossover, we would not distinguish another class for them.

Note that the geometric crossover requires a way to appoint a segment between parent semantics. This imposes important restrictions to the representation of semantics, as it must be an object in a normed vector space. However if this condition is met, an (exact) geometric crossover is guaranteed to breed offspring that is not worse, than the worst of its parents [18].

In this group, special attention should be paid to *semantic geometric crossover* (SGX) by Moraglio et al. [25], which guarantees that the semantics of an offspring lies on the segment spanned over semantics of its parents, i.e., SGX is exact geometric crossover. Unfortunately SGX causes exponential in time bloat, which puts its practical applications into question, even when using implementation techniques that allow to handle exponentially increasing in time program structures in linearly-increasing data structures [5].

The next operator that is worth to mention is *approximately geometric semantic crossover* by Krawiec and Lichocki (KLX) [16]. KLX is indeed a meta-operator that runs in a loop some other (secondary) crossover operator, and from all the offspring produced by the secondary operator, chooses the most geometric according to an arbitrary geometry measure.

Algorithm 1. Krawiec Lichocki Crossover (KLX). p_1, p_2 are parents, k is a number of crossover attempts, SX is a secondary crossover, MOSTGEOMETRIC is Eq. 1.

```

1: function KLX( $p_1, p_2, k$ )
2:    $O \leftarrow \emptyset$  ▷ Offspring candidates
3:   for all  $i \in 1..k$  do
4:      $O \leftarrow O \cup \text{SX}(s(p_1), s(p_2))$ 
5:    $o_1 \leftarrow \text{MOSTGEOMETRIC}(O, p_1, p_2)$ 
6:    $o_2 \leftarrow \text{MOSTGEOMETRIC}(O \setminus \{o_1\}, p_1, p_2)$ 
7:   return  $\{o_1, o_2\}$ 
8: end function
    
```

Other operators in this group are: *locally geometric semantic crossover* [19,17], that approximates geometric recombination of parents at the level of the homologous crossover point, and *approximately geometric semantic crossover* [27,18], that propagates the desired geometric semantics of offspring back through the tree of parent program, in order to place an adequate subtree at the level of crossover point.

3 Semantically-Effective Geometric Crossover Operator

One may ask, whether is it possible to combine features of semantically-effective and geometric crossover operators to benefit from advantages of both of them?

To achieve that we can equip almost any geometric crossover operator, with a procedure that prevents the operator from producing semantically ineffective offspring w.r.t. its parents.

To be more specific, we propose such a procedure for SX+ variant of KLX operator described in [16]. The original KLX algorithm is shown in Algorithm 1. It operates by running k times a secondary crossover operator, e.g., tree-swapping crossover [13], and finally choosing the most geometric offspring that it produced, according to the formula:

$$\text{MOSTGEOMETRIC}(O, p_1, p_2) = \arg \min_{o \in O} \underbrace{d(s(p_1), s(o)) + d(s(o), s(p_2))}_{\text{distance sum}} + \underbrace{|d(s(p_1), s(o)) - d(s(o), s(p_2))|}_{\text{penalty}} \quad (1)$$

where O is a set of candidate offspring, p_1, p_2 are parent programs, $s(p)$ is semantics of program p , $d(\cdot, \cdot)$ is a distance metric. The formula consists of two major parts: the first one is sum of distances between the semantics of candidate offspring o and both parents; the second one is a penalty for choosing offspring that is not equidistant from parents.

To combine KLX with principles of semantically effective crossover, we propose to replace MOSTGEOMETRIC calls in Algorithm 1 with the formula:

$$\text{MOSTGEOMETRIC}^+(O, p_1, p_2) = \begin{cases} \text{MOSTGEOMETRIC}(O', p_1, p_2) & O' \neq \emptyset \\ \text{MOSTGEOMETRIC}(O, p_1, p_2) & O' = \emptyset \end{cases} \quad (2)$$

where $O' = \{o : o \in O, s(o) \neq s(p_1), s(o) \neq s(p_2)\}$

Table 1. Parameters of evolution

Parameter	Value
Population size	1024
Fitness function	<i>Symbolic regression:</i> Mean absolute error <i>Boolean domain:</i> Hamming distance
Termination condition	At most 100 generations or find of individual having fitness 0
Initialization method	Ramped Half-and-Half, height range 2 – 6, up to 100 retries
Selection method	Tournament selection, size 7
Max program height	17
Crossover probability	1.0
Number of attempts	KLX ⁺ , KLX, SAC, GPX ⁺ : 10, GPX: n/a
Semantic sensitivity	10 ⁻⁶
Instructions	<i>Symbolic regression:</i> x, +, -, ×, /, sin, cos, exp, log, ERC ^a <i>Boolean domain:</i> D1...D11 ^b , and, or, nand, nor, ERC
Number of runs	30

^a log and / are protected. log is defined as log |x|; / returns 0 if divisor is 0.

^b Number of inputs depends on a problem instance.

Table 2. Benchmarks; in symbolic regression there are 20 *equidistant* fitness cases in the given range

Symbolic regression			Boolean domain		
Problem	Definition (formula)	Range	Problem	Instance (bits)	Fitness cases
Septic	$x^7 - 2x^6 + x^5 - x^4 + x^3 - 2x^2 + x$	[-1, 1]		PAR5	32
Nonic	$\sum_{i=1}^9 x^i$	[-1, 1]	Even parity	PAR6	64
R1	$(x + 1)^3 / (x^2 - x + 1)$	[-1, 1]		PAR7	128
R2	$(x^5 - 3x^3 + 1) / (x^2 + 1)$	[-1, 1]	Multiplexer	MUX6	64
R3	$(x^6 + x^5) / (x^4 + x^3 + x^2 + x + 1)$	[-1, 1]		MUX11	2048
Nguyen6	$\sin(x) + \sin(x + x^2)$	[-1, 1]	Majority	MAJ6	64
Nguyen7	$\log(x + 1) + \log(x^2 + 1)$	[0, 2]		MAJ7	128
Keijzer1	$0.3x \sin(2\pi x)$	[-1, 1]	Comparator	CMP6	64
Keijzer4	$x^3 e^{-x} \cos(x) \sin(x) (\sin^2(x) \cos(x) - 1)$	[0, 10]		CMP8	256

This change restricts the set of candidate offspring by removing offspring that is semantically equal to any of the parents. Only if all candidates are semantically equal, the algorithm uses the whole candidate set. For semantics represented by floating point numbers, it is natural to compare semantics in Eq. 2 with a threshold. To be consistent with [30], we call it *semantic sensitivity*. We denote the augmented variant of KLX as KLX⁺.

4 The Experiment

We attempt to experimentally verify whether combining features of geometric and semantically-effective crossover operators pays off. In order to do that we prepare a run of GP employed with KLX⁺ and we compare it to GP running ‘bare’

Table 3. Average fitness and 95% confidence interval achieved by best-of-run individual as of 100 generation. Best values are marked in bold.

Problem	KLX ⁺	KLX	SAC	GPX ⁺	GPX
Keijzer1	0.005 ±0.002	0.015 ±0.003	0.008 ±0.003	0.009 ±0.003	0.013 ±0.004
Keijzer4	0.030 ±0.011	0.139 ±0.014	0.049 ±0.013	0.045 ±0.014	0.041 ±0.010
Nguyen6	0.001 ±0.000	0.005 ±0.002	0.002 ±0.001	0.002 ±0.001	0.004 ±0.002
Nguyen7	0.001 ±0.000	0.005 ±0.002	0.002 ±0.001	0.003 ±0.001	0.005 ±0.002
Nonic	0.006 ±0.002	0.024 ±0.003	0.012 ±0.003	0.014 ±0.004	0.018 ±0.004
Septic	0.013 ±0.005	0.063 ±0.026	0.020 ±0.004	0.027 ±0.009	0.032 ±0.008
R1	0.006 ±0.002	0.028 ±0.006	0.012 ±0.003	0.022 ±0.006	0.022 ±0.005
R2	0.015 ±0.004	0.028 ±0.005	0.017 ±0.004	0.021 ±0.005	0.028 ±0.007
R3	0.001 ±0.000	0.006 ±0.001	0.002 ±0.000	0.003 ±0.001	0.003 ±0.001
CMP6	0.000 ±0.000	0.533 ±0.221	0.267 ±0.158	0.100 ±0.107	0.867 ±0.303
CMP8	0.067 ±0.128	7.300 ±0.956	7.000 ±1.070	6.167 ±0.570	9.433 ±1.182
MAJ6	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000	0.000 ±0.000
MAJ7	0.000 ±0.000	0.133 ±0.153	0.100 ±0.142	0.367 ±0.269	0.567 ±0.272
MUX6	1.700 ±0.717	5.667 ±0.642	3.133 ±0.732	3.200 ±0.770	3.567 ±0.683
MUX11	102.567 ±12.470	160.067 ±14.555	114.000 ±8.641	118.467 ±7.202	118.467 ±7.048
PAR5	0.000 ±0.000	2.033 ±0.510	3.500 ±0.604	2.433 ±0.440	3.400 ±0.567
PAR6	5.000 ±0.947	13.367 ±0.792	11.567 ±1.204	13.167 ±0.795	14.200 ±0.893
PAR7	23.467 ±1.044	40.967 ±1.047	35.967 ±1.752	40.533 ±0.799	41.033 ±1.899

KLX. In addition we add two control setups: tree-swapping crossover (GPX) [13], which acts as a reference point, and GPX⁺ – GPX that rejects semantically ineffective offspring and retries crossover act until it is effective or a given number of attempts is exceeded. We added GPX⁺ to check whether the achievements of KLX⁺ are due to the combination of properties of geometric and semantically-effective crossovers, or solely due to restrictions for ineffective offspring. Note that GPX⁺ is not equivalent to SAC [30], as the latter one performs equivalence check in the crossover points instead of whole program trees, i.e., SAC allows an exchange of semantically different subtrees, however it does not take into account, how this exchange influences the semantics of entire trees, which could not change at all. However, as an previously published operator, SAC is good reference point, therefore we add it as the last setup. Details of evolutionary parameters can be found in Table 1, parameters not included there, are set to ECJ’s defaults [22]. Note that the experiment does not include any mutation operator, since it is focused on the examination of properties of crossover operators.

We run evolution of 18 commonly used benchmark problems shown in Table 2: 9 symbolic regression [13,23] and 9 Boolean function synthesis [13,33] benchmarks.

Performance

Figure 1 presents the plots of average and 95% confidence interval of the best-of-generation fitness achieved by each operator and Table 3 shows average and 95% confidence interval of the best-of-run fitness.

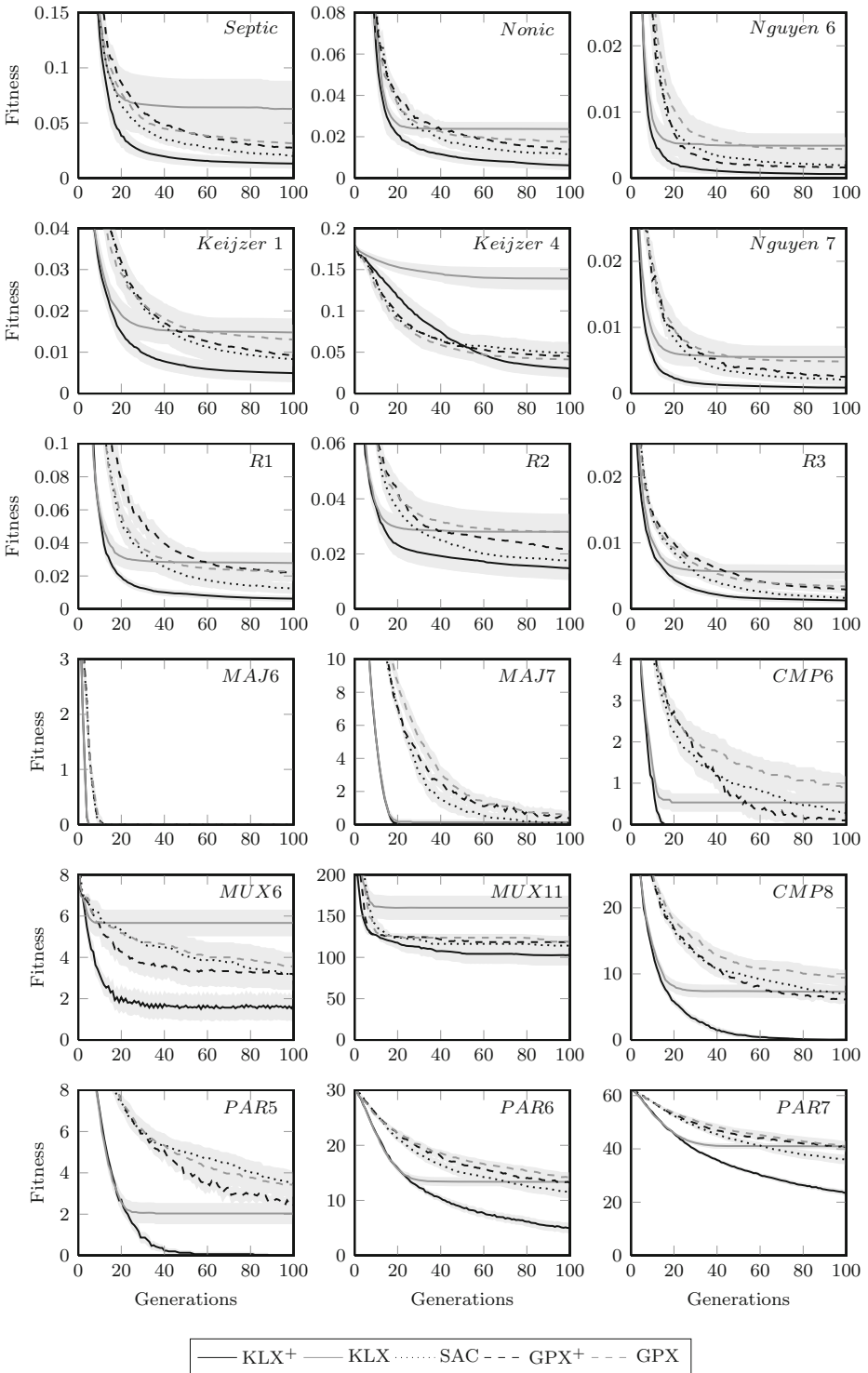


Fig. 1. Fitness achieved by best-of-generation individual averaged over 30 runs. Shadings are 95% confidence intervals.

Table 4. Post-hoc analysis of Friedman’s test using symmetry test: p-values of incorrectly judging operator in row as outranking operator in column. Significant p-values are marked in bold ($\alpha = 0.05$) and visualized as arcs in outranking graph.

	GPX	GPX ⁺	KLX	KLX ⁺	SAC
GPX			0.976		
GPX ⁺	0.151		0.033		
KLX					0.052
KLX ⁺	0.000	0.004	0.000		
SAC	0.017	0.926	0.002		

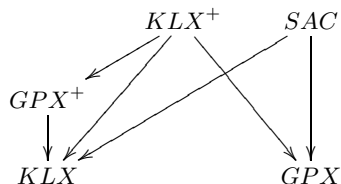


Table 5. Fraction and 95% conf. interval of ineffective crossover acts (the lowest in bold)

Domain	KLX ⁺	KLX	SAC	GPX ⁺	GPX
Symbolic regression	0.001 ± 0.0000	0.883 ± 0.0001	0.067 ± 0.0001	0.002 ± 0.0000	0.113 ± 0.0001
Boolean domain	0.000 ± 0.0000	0.902 ± 0.0001	0.649 ± 0.0002	0.001 ± 0.0000	0.525 ± 0.0002

It is clear that KLX⁺ converges the quickest and in all benchmark problems achieves the best fitness at the end of evolution. Moreover, thanks to quick convergence, KLX⁺’s fitness achieved after 30 generations, in 15 out of 18 benchmarks remains unbeatable by any other operator at the end of evolution.

To verify statistical significance of obtained results, we performed Friedman’s test for multiple achievements of multiple subjects [11] on data shown in Table 3. The calculated p-value is 6.52×10^{-9} , thus assuming critical value $\alpha = 0.05$, the test is conclusive, that there is at least one statistical difference in the results. Therefore we present in Table 4 a post-hoc analysis of Friedman’s test, using the symmetry test, and an outranking graph for the operators.

The analysis shows that KLX⁺ outranks all other operators except SAC, however the p-value for outranking SAC, i.e., 0.052, is very close to the critical value of $\alpha = 0.05$. From the graph we can see that semantically-effective SAC outranks GPX and KLX, and semantically-effective GPX⁺ outranks KLX only. Perhaps the low efficiency of KLX can be surprising, however we believe that KLX is likely to stick in local optima due to lack of routine that prevents from producing ineffective offspring, that exists in superior KLX⁺. This leads us to conclusion that rejection of semantically-ineffective offspring is an important factor that influences GP performance.

Semantic Effectiveness

An observant reader may ask why semantic-effectiveness has such a significant influence on operator’s performance. To answer this question, we presented in Table 5 fraction of ineffective crossover acts done by each operator separately during all evolutionary runs of previous experiment.

Values obtained by KLX are clearly the highest and show that nearly 90% of crossover acts done by KLX are actually ineffective. We believe that this is

mainly caused by bias of Eq. 1, which prefers offspring having the same semantics as a parent over the offspring that is nearly-distant from one of the parents, but non-geometric². The high fraction of ineffective offspring clearly explains the observed premature convergence of KLX.

On the other hand SAC maintains low percentage of ineffective offspring, however only for symbolic regression. We hypothesize that high value for Boolean domain is due to our previous observation, that SAC performs equivalence check only at the level of crossover point, and since Boolean instructions are likely to return unchanged output if only one input changes, it likely makes the crossover operation ineffective at higher parts of program tree. This seems to be consistent with results of GPX⁺, which are very low for both domains, and significantly lower than its unbiased counterpart – GPX.

5 Conclusions

We divided semantic crossover operators for tree-based GP into two classes depending on the purpose of using program semantics: to reject semantically ineffective offspring, or to geometrically combine parent programs. Moreover we proposed a way to combine features of these two classes in a single operator and experimentally demonstrated that this combination performs better and is less likely to suffer from premature convergence than each approach solely. We believe that the proposed method can be successfully applied to other geometric semantic crossover operators as well.

Acknowledgment. We would like to thank Wojciech Jaśkowski and Krzysztof Krawiec for their time and the valuable comments on this study. Work supported by Polish National Science Centre grant no. DEC-2012/07/N/ST6/03066.

References

1. Al-Sakran, S.H., Koza, J.R., Jones, L.W.: Automated re-invention of a previously patented optical lens system using genetic programming. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 25–37. Springer, Heidelberg (2005)
2. Beadle, L., Johnson, C.: Semantically driven crossover in genetic programming. In: IEEE CEC 2008, pp. 111–116. IEEE Press (2008)
3. Beadle, L., Johnson, C.G.: Semantic analysis of program initialisation in genetic programming. *Genetic Programming and Evolvable Machines* 10(3), 307–337 (2009)
4. Beadle, L., Johnson, C.G.: Semantically driven mutation in genetic programming. In: IEEE CEC 2009, pp. 1336–1342. IEEE Press (2009)

² Eq. 1 evaluates to $2d(s(p_1), s(p_2))$ for offspring semantically equal to any of its parents.

5. Castelli, M., Castaldi, D., Giordani, I., Silva, S., Vanneschi, L., Archetti, F., Maccagnola, D.: An efficient implementation of geometric semantic genetic programming for anticoagulation level prediction in pharmacogenetics. In: Correia, L., Reis, L.P., Cascalho, J. (eds.) EPIA 2013. LNCS, vol. 8154, pp. 78–89. Springer, Heidelberg (2013)
6. D'haeseleer, P.: Context preserving crossover in genetic programming. In: IEEE CEC 1994, vol. 1, pp. 256–261. IEEE Press (1994)
7. Ferreira, C.: Genetic representation and genetic neutrality in gene expression programming. *Advances in Complex Systems* 5(4), 389–408 (2002)
8. Galvan-Lopez, E., et al.: Using semantics in the selection mechanism in genetic programming: A simple method for promoting semantic diversity. In: IEEE CEC 2013, vol. 1, pp. 2972–2979 (2013)
9. Harries, K., Smith, P.: Exploring alternative operators and search strategies in genetic programming. In: GP 1997, pp. 147–155. Morgan Kaufmann (1997)
10. Jackson, D.: Phenotypic diversity in initial genetic programming populations. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 98–109. Springer, Heidelberg (2010)
11. Kanji, G.: 100 Statistical Tests. SAGE Publications (1999)
12. Keller, R.E., Banzhaf, W.: Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In: GP 1996, pp. 116–122. MIT Press (1996)
13. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
14. Koza, J.R., et al.: Genetic Programming 3: Darwinian Invention and Problem Solving. Morgan Kaufman (April 1999)
15. Koza, J.R., et al.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers (2003)
16. Krawiec, K., Lichocki, P.: Approximating geometric crossover in semantic space. In: GECCO 2009, pp. 987–994. ACM (2009)
17. Krawiec, K., Pawlak, T.: Quantitative analysis of locally geometric semantic crossover. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 397–406. Springer, Heidelberg (2012)
18. Krawiec, K., Pawlak, T.: Approximating geometric crossover by semantic back-propagation. In: GECCO 2013, pp. 941–948. ACM (2013)
19. Krawiec, K., Pawlak, T.: Locally geometric semantic crossover: A study on the roles of semantics and homology in recombination operators. *Genetic Programming and Evolvable Machines* 14(1), 31–63 (2013)
20. Langdon, W.B.: Size fair and homologous tree genetic programming crossovers. *Genetic Programming and Evolvable Machines* 1(1/2), 95–119 (2000)
21. Lohn, J., Hornby, G., Linden, D.: An evolved antenna for deployment on Nasa's Space Technology 5 Mission. In: Genetic Programming Theory and Practice II, ch. 18, pp. 301–315. Springer (2004)
22. Luke, S.: The ECJ Owner's Manual – A User Manual for the ECJ Evolutionary Computation Library, zeroth edition, online version 0.2 edition (October 2010)
23. McDermott, J., et al.: Genetic programming needs better benchmarks. In: GECCO 2012, pp. 791–798. ACM (2012)
24. Moraglio, A.: Abstract convex evolutionary search. In: FOGA XI, pp. 151–162. ACM (2011)

25. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 21–31. Springer, Heidelberg (2012)
26. Nguyen, Q.U., Nguyen, X.H., O’Neill, M.: Semantics based mutation in genetic programming: The case for real-valued symbolic regression. In: Mendel 2009, pp. 73–91 (2009)
27. Pawlak, T.P., Wieloch, B., Krawiec, K.: Semantic backpropagation for designing search operators in genetic programming. *IEEE Transactions on Evolutionary Computation* (2014)
28. Poli, R., Langdon, W.B.: Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation* 6(3), 231–252 (1998)
29. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Lulu Enterprises, UK Ltd. (2008)
30. Uy, N.Q., et al.: Semantically-based crossover in genetic programming: Application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines* 12(2), 91–119 (2011)
31. Uy, N.Q., et al.: On the roles of semantic locality of crossover in genetic programming. *Information Sciences* 235, 195–213 (2013)
32. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines* (online first)
33. Walker, J.A., Miller, J.F.: Investigating the performance of module acquisition in cartesian genetic programming. In: GECCO 2005, vol. 2, pp. 1649–1656. ACM Press (2005)

On the Locality of Standard Search Operators in Grammatical Evolution

Ann Thorhauer and Franz Rothlauf

University of Mainz, Germany
{thorhauer, rothlauf}@uni-mainz.de
<http://www.uni-mainz.de>

Abstract. Offspring should be similar to their parents and inherit their relevant properties. This general design principle of search operators in evolutionary algorithms is either known as locality or geometry of search operators, respectively. It takes a geometric perspective on search operators and suggests that the distance between an offspring and its parents should be less than or equal to the distance between both parents. This paper examines the locality of standard search operators used in grammatical evolution (GE) and genetic programming (GP) for binary tree problems. Both standard GE and GP search operators suffer from low locality since a substantial number of search steps result in an offspring whose distance to one of its parents is greater than the distance between both of its parents. Furthermore, the locality of standard GE search operators is higher than that of standard GP search operators, which allows more focused search in GE.

Keywords: Grammatical evolution, genetic programming, locality, geometric crossover, random walk.

1 Introduction

Recombination operators in evolutionary algorithms (EA) aim to construct offspring solutions in such a way that the offspring inherit the properties of their parents. Thus, offspring are constructed using genetic material of their parents, in order to ensure that they are “similar” to them. Analogously, the mutation operators in EA modify the offspring solutions so that the new “mutated” solution is only slightly different from the original solution. Recombination and mutation operators thus both follow the principle of creating solutions that are similar to the original solution. Violating this principle would result in random search since, in this case, the offspring would be highly dissimilar to their parents, and the evolutionary search process would not be able to focus on promising areas of the search space [4].

This basic principle of genetic search operators was first formulated by Liepins and Vose [13] and Radcliffe [17,22], who recognized that search operators cannot be designed independently of the search space. On the contrary, their design must be based on the metric defined in the search space. Indeed, mutation should

create an offspring x^o from a parent x^p in such a way that the distance $d(x^p, x^o)$ between parent and offspring is low. Analogously, given two parental solutions x_1^p and x_2^p and one offspring solution x^o , recombination operators should be designed in such a way that $\max(d(x_1^p, x^o), d(x_2^p, x^o)) \leq d(x_1^p, x_2^p)$ [17][18, p. 62], which means that the distances between offspring and parents should be less than or equal to the distance between the parents. By viewing the distance between two solutions as a measurement of dissimilarity, this design principle ensures that the offspring solutions are similar to the original (parent) solution.

This general design principle of search operators introduced by Liepins and Radcliffe [13,17] was later denoted by Rothlauf as the “locality of search operators” [18,19] and by Moraglio as the “geometry of search operators” [15,14]. Mutation operators have high locality, i.e., are geometric if offspring solutions are similar to their parents; analogously, recombination operators have high locality, i.e., are geometric if the distances between offspring and parents are less than or equal to the distance between the parents. Crossover and mutation are defined representation-independent using the notion of distance associated with the search space. The geometric terms use the notions of line segment and ball, which are well defined once a notion of distance in the search space is defined [15].

This paper studies the locality of standard search operators used in GE (crossover, mutation, and duplication) and GP (crossover and reproduction). We examined whether the GE and GP search operators have high locality, i.e., are geometric. In the experiments, we focused on binary trees and performed random walks through the binary tree search space by measuring distances between both parents x_1^p and x_2^p as well as between an offspring x^o and its parents. The locality of search operators is high if $\max(d(x_1^p, x^o), d(x_2^p, x^o)) \leq d(x_1^p, x_2^p)$.

In Sect. 2, we define the locality of search operators and provide a brief overview of the literature on locality. In Sect. 3, we present the experiments and results. The paper ends with some concluding remarks.

2 Locality of Search Operators

Each search space can be defined as a *topological space*, which describes similarities between solutions by defining the relationships between sets of solutions. Formally, a topological space is an ordered pair (X, T) , where X is a set of solutions and T is a collection of subsets of X called open sets. We can define different topologies (search spaces) by combining X with different T . For example, *metric search spaces* are a specialized form of topological spaces where similarities between solutions are measured by a distance. In metric search spaces, we have a set X of solutions and a real-valued distance function (also called a metric) $d : X \times X \rightarrow \mathbb{R}$ that assigns a real-valued distance to any combination of two elements $x, y \in X$.

The locality of search operators [18] is equivalent to the concept of geometry of search operators [15]. Both define search operators based on the metric of the search space. Given a metric, we are able to define distances between solutions.

In particular, a mutation operator has high locality if the distance between the resulting offspring and its parent is small. From a geometric perspective, the offspring are in the space-specific ball of a small radius centered in the parent. Analogously, a crossover operator has high locality if the distance between offspring x^o and its parents is less than or equal to the distance between both parents x_1^p and x_2^p ($\max(d(x_1^p, x^o), d(x_2^p, x^o)) \leq d(x_1^p, x_2^p)$); using the notion of geometry, a crossover is geometric if all offspring are in the space-specific segment between their parents [15].

We want to emphasize that the locality of search operators is different from the locality of representations [18]. Representations are genotype-phenotype mappings that assign genotypes to phenotypes. In both search spaces (genotype and phenotype space), a metric defines distances between solutions. However, the metric used in the genotype space and the phenotype space can be different. The locality of a representation describes how well the distances between genotypes fit the distances between the corresponding phenotypes. Thus, a representation has high locality if distances between genotypes are similar to the corresponding phenotype distances, for example if neighboring genotypes correspond to neighboring phenotypes. Analogously, a representation has low locality if genotype and phenotype distances do not fit together, for example if neighboring genotypes are not neighbors in the phenotype search space. Although both concepts, the locality of search operators and the locality of a representation, are based on the notion of distance between solutions, they are quite different and should not be confused with one another. The locality of a representation is relevant for the design of representations, whereas the locality of search operators for the design of meaningful search operators.

2.1 Locality in Genetic Programming

There are a number of studies on the locality of GP [5,6,7], however they do not examine the locality of search operators nor the locality of representations; rather, they focus on the locality of the genotype-fitness mapping. They are mainly interested in how the choice of mutation operators affects the changes in the corresponding fitness values. For example, Galván-López et al. [7] study genotype-fitness mappings and find “that the mutation operators examined are inconsistent with respect to the quality of locality as measured by fitness and structural changes”. In a similar paper, Galván-López et al. [5] find that “when the original fitness is low, large genotype jumps can lead to fitness improvements” [5]. In contrast, “when the original fitness is high, large [genotype mutations] tend to be quite detrimental” [5]. Galván-López et al. [6] study whether small genotype changes correspond to small fitness changes. The authors introduce different neighborhood functions on the fitness space and observe that their ability to serve as good predictor of GP performance is limited.

Another study by Uy and his co-workers distinguishes between syntactic and semantic locality of crossover in GP [23,24,25]. They notice that “most GP genetic operators have been designed based on syntax alone; but small changes in syntax can lead to large changes in semantics.” [24]. The authors introduce a

new semantic similarity based crossover (SSC), which ensures that exchanged subtrees between individuals are semantically similar concerning fitness values. SSC leads to higher GP performance since the resulting genotype step size is smaller than in standard GP operators. Uy et al. [23] compare the syntactic crossover in GP with the semantic crossover in GP. With locality measuring the differences between two subtrees, they find that syntactic locality (measured using Levenshtein tree distance) is less important than semantic locality (measured using fitness differences).

In summary, it is still unclear whether standard GP search operators have high locality, i.e., are geometric. There is evidence that small genotype changes can also lead to large fitness changes, which is detrimental for guided search in GP.

2.2 Locality in Grammatical Evolution

GE [16] is a variant of GP that can evolve complete programs in an arbitrary language using a variable-length binary string. In GE, phenotype expressions are created from binary genotypes by using a complex genotype-phenotype mapping. A genotype consists of groups of eight bits (denoted as codons) which encode an integer value that selects production rules from a grammar in BNF. These rules are used in the mapping process to create a phenotype. The mapping process is deterministic since the same genotype always results in the same phenotype.

The standard GE recombination operator [16] is similar to the cut and splice operator introduced in [8]. After selecting two parents, a crossover point is randomly selected for each parent. Then, the genetic material beyond these points is exchanged between the parents. As a result, rather than remain constant, the length of the genotype changes during the search. The standard GE mutation operator [16] randomly changes the integer value of a codon. The third standard GE operator, duplication, increases the number of available genetic material. It copies a random number of codons starting at a randomly selected start codon and inserts them between the second last and last codon in the genotype.

There are no studies on the locality of GE search operators. Instead, existing work focuses on the locality of the genotype-phenotype mapping or on how search performance depends on the type of search operator. Rothlauf and Oetzel [20] study the locality of the genotype-phenotype mapping and find that “the representation used in GE has problems with locality as many neighboring genotypes do not correspond to neighboring phenotypes.” Byrne et al. [2,1] distinguish between two types of mutation operators in GE: the structural mutation (that changes the shape of the derivation tree) and the high-locality nodal mutation (that changes the value of a node). They examine the impact of these operators on search performance and find out that both have different goals in a GE search process: exploration and exploitation. Castle and Johnson [3] study the mutation and crossover points in GE and find “that events occurring at the first positions of a genotype are indeed more destructive, but also indicate that they may be the most constructive crossover and mutation points” [3]. Finally, Hugosson et al. [9] examine different binary-integer representations (Gray versus

binary code) in GE. They find that the choice of the binary-integer mapping has no influence on GE search performance.

3 Experiments and Results

We studied the locality of standard search operators used in GE and GP. Standard GE search operators are (one-point) crossover, (integer) mutation, and duplication [21]. Standard GP search operators are crossover and reproduction [11]. The locality of recombination operators is high if the distances between offspring and parents are less than or equal to the distance between both parents. For mutation and duplication, high locality implies a low distance between offspring and parent.

3.1 Experimental Design

To study the locality of search operators, we performed random walks using different types of GE and GP search operators. We did not use a selection operator. The search operators created two offspring x_1^o and x_2^o from two parents x_1^p and x_2^p , which replaced their parents. If the variation operators included recombination, we measured the distance $d(x_1^p, x_2^p)$ between both parents as well as the distances $d(x_i^o, x_j^p)$ ($i, j \in \{1, 2\}$) between each of the two offspring and their two parents. If we applied only mutation or duplication (and no crossover), we created one offspring x^o from each of the two parents. Each offspring replaced the corresponding parent. To evaluate the locality of mutation and duplication, we measured the distance $d(x^o, x^p)$ between each offspring and its corresponding parent.

For both GE and GP, the definition of the terminal and function set is relevant for the distances between solutions. In the current study, we focused on problems where solutions are binary trees. Thus, the number of terminals $|T| = 1$ equals the number of functions $|F| = 1$. For GE, we used two production rules: the first one chose between a binary function and a terminal (e.g. $\langle expr \rangle ::= \langle expr \rangle + \langle expr \rangle \mid \langle var \rangle$) and the second one defined a terminal (e.g. $\langle var \rangle ::= X$). The fitness of binary trees using associative binary functions (like $+$, $-$, $*$, $/$) is determined only by its size l (number of terminals plus number of functions) since the order of traversing such a tree is irrelevant (due to the associativity of the function). For example, for $T = \{x\}$ and $F = \{+\}$, all feasible trees of size l encode the expression $(l + 1) * \frac{x}{2}$. Consequently, we measured the distance between two solutions by using the Levenshtein distance as metric, that is, the minimal number of operations that are needed to transform one expression into another [12] between the two encoded expressions. For the example ($T = \{x\}$ and $F = \{+\}$), the Levenshtein distance between two valid binary trees x_i and x_j of length l_i and l_j is equal to $|l_i - l_j|$. We should be aware that the size l of a valid tree is always odd. Furthermore, for two valid binary trees, all possible distances are even. In contrast to GP, GE search operators can also create invalid solutions, where the genotype-phenotype mapping process cannot be finished.

In this case, the Levenshtein distance between the two corresponding expressions can be odd since the size of invalid expressions can be even.

In the GP experiments, we applied no constraint on the maximum allowable tree depth. For GE, we did not use a wrapping operator since for all the solutions (binary trees) where wrapping would be necessary the mapping process would never terminate and thus the corresponding individuals would be invalid anyway. Each random walk started with two random, yet identical, solutions. For GP, the initial solutions were created using the grow method with a maximum depth $d_{max} = 6$. For GE, the initial solutions were randomly created with a maximum length of 10 codons (80 binary alleles).

Throughout the random walk, the variation operators were applied with standard probabilities. For GP, the crossover probability was $p_c = 0.9$ (biased towards selecting internal nodes with a probability of 0.9) and the reproduction probability was $p_r = 0.1$ [10]. For GE, the crossover probability was $p_c = 0.9$, the mutation probability was $p_m = 0.01$, and the duplication probability was $p_d = 0.01$ [21]. In each search step, two new offspring that replaced their parents were generated. Each random walk terminated after 50 search steps, generating overall 100 offspring. For each experimental setting, we performed 100,000 random walks resulting in a total of 10 million offspring.

3.2 Results

For GE and GP, we studied the locality of the combined standard search operators as well as the locality of recombination alone. Furthermore, we examined the locality of the mutation and duplication operator used in GE only.

Locality of Standard Search Operators. We studied whether the standard search operators used in GE (crossover, mutation, duplication) and GP (crossover, reproduction) have high locality. In each step of the random walk, the GE operators were applied with probabilities $p_c = 0.9$, $p_m = 0.01$, and $p_d = 0.01$ and the GP operators were applied with $p_c = 0.9$ and $p_r = 0.1$.

For GE, in 75.4% of all cases, the minimal distance $\min(d(x_1^p, x^o), d(x_2^p, x^o))$ between an offspring and its parents is equal to 0. For GP, 54.5% of all offspring are identical to at least one of its parents ($\min(d(x_1^p, x^o), d(x_2^p, x^o)) = 0$). In the following plots, we will ignore all such applications of search operators.

Figure 1 plots the distribution of $d(x^o, x_j^p)$ ($j \in \{1, 2\}$) over $d(x_1^p, x_2^p)$. For increased clarity, we only show the results for $d(x_1^p, x_2^p), d(x^o, x^p) \leq 20$. For a given distance $d(x_1^p, x_2^p)$, the gray-coded squares indicate the percentage of offspring whose distance to one of their parents is equal to $d(x^o, x^p)$ (darker squares indicate a higher percentage of offspring). For example, for GE and parental distance $d(x_1^p, x_2^p) = 4$, about 22.8% of all offspring have $d(x^o, x_j^p) = 1$, 31.8% have $d(x^o, x_j^p) = 2$, 25% have $d(x^o, x_j^p) = 3$, 0% have $d(x^o, x_j^p) = 4$ (they are duplicates of one parent and thus excluded from analysis), 9.8% have $d(x^o, x_j^p) = 5$, and so on. Search operators have high locality if they produce only offspring with $d(x^o, x_i^p) \leq d(x^{p1}, x^{p2})$ ($i \in \{1, 2\}$), which are located in the lower right triangle of the plot below the line through origin. All offspring

located in the upper left triangle are the result of a low-locality operator. The plots indicate that standard search operators of GE as well as GP suffer from low locality since a substantial number of random walk steps resulted in offspring whose distance to one of their parents is greater than the distance between both parents. We should be aware that there are only even-numbered distances between GP solutions (leading to “holes” in the GP plot), whereas for GE odd-numbered distances also exist (either the parent or the offspring is invalid).

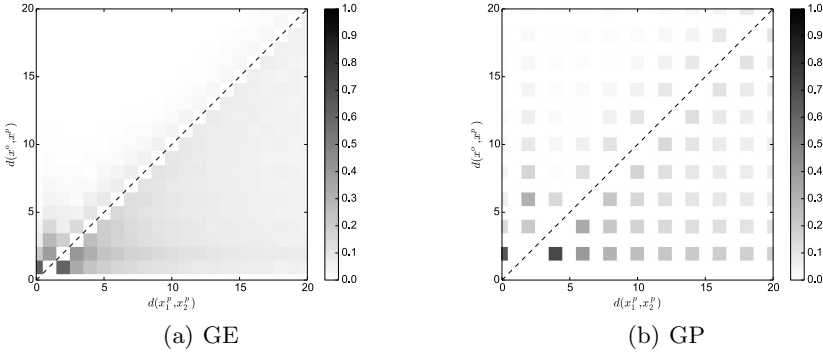


Fig. 1. Distribution of $d(x^o, x_j^p)$ ($j \in \{1, 2\}$) over $d(x_1^p, x_2^p)$

To get deeper insights into the locality of standard search operators, we will now focus on all offspring that are generated from parents with a given distance $d(x^{p1}, x^{p2})$. Figure 2 plots the number of offspring x^o (cumulative relative frequency) over $d(x^o, x_j^p)$ ($j \in \{1, 2\}$) for fixed distances $d(x^{p1}, x^{p2}) \in \{0, 4, 8, 12, 16, 20\}$. Each line represents a vertical cut through Fig. 1, summing up all offspring whose distance to its parents is less than or equal to $d(x^o, x^p)$. For example, for GE and parental distance $d(x_1^p, x_2^p) = 4$, 79.6% of all offspring have a distance to their parents that is less than or equal to 4 ($d(x^o, x_j^p) \leq 4$). Thus, in 79.6% cases the standard GE search operators have high locality. Search operators would have perfect locality if the cumulative frequency was 1 for $d(x^o, x^p) \leq d(x_1^p, x_2^p)$. Table 1 summarizes the percentage of applications of standard GE and GP search operators resulting in an offspring where $d(x^o, x^p) > d(x_1^p, x_2^p)$.

We see that standard GE and GP operators suffer from low locality since a substantial number of offspring display a distance to one of their parents that is greater than the distance between their parents. In general, the locality of standard GE search operators is higher than that of standard GP operators. For example, for $d(x_1^p, x_2^p)=8$, about 90% of all GE offspring but only about 75% of all GP offspring have less or equal distances to their parents than their parents do to each other.

Table 1. Percentage of offspring, where $d(x^o, x^p) > d(x_1^p, x_2^p)$

$d(x_1^p, x_2^p)$	4	8	12	16	20
GE	20.4	11.2	7.1	7.1	6.5
GP	28.2	25.3	23.2	21.6	20.3

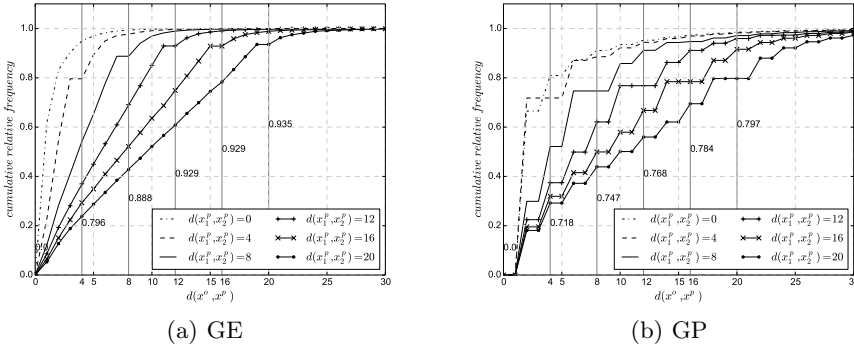


Fig. 2. Number of offspring x^o (cumulative frequency) over $d(x^o, x^p)$ for fixed distances $d(x_1^p, x_2^p) \in \{0, 4, 8, 12, 16, 20\}$

Locality of GE Crossover. We studied the locality of the GE and GP crossover operator. When using the same experimental design as above, we were faced with the problem that performing a GE random walk with only crossover would lead to a non-representative sample of offspring. Since crossover alone cannot increase the genetic material of the genotypes (this is the aim of duplication), it would only reshuffle genetic material between the two random walk solutions and not create representative GE solutions obtained in a standard GE run. Thus, to ensure representative GE solutions and to also be able to generate also longer GE genotypes, we slightly modified our experimental setting. We considered all 10 million offspring created in the random walks using the combined standard search operators as described above, but only applied crossover with $p_c = 1$ to each pair of offspring. By only applying crossover to the solutions generated by standard search operators, we were able to study the locality of crossover in detail.

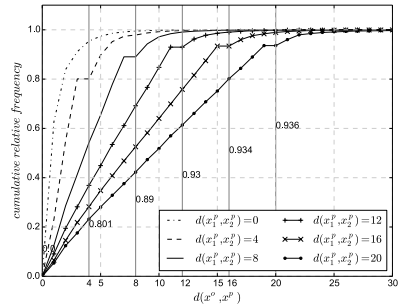


Fig. 3. GE with only crossover: number of offspring x^o (cumulative frequency) over $d(x^o, x^p)$ for fixed distances $d(x_1^p, x_2^p)$

We will only present results for GE, since the results for GP are identical to Figs. 1(b) and 2(b). Since the GP reproduction operator just copies a parent to its offspring, the locality of crossover plus reproduction is equal to crossover alone. Figure 3 shows the results for GE using only crossover ($p_c = 1$, no mutation or duplication). The comparison of these results to the previous results of the combined standard search operators (Fig. 2(a)) reveals no larger differences.

Thus, the locality of GE standard search operators is mainly determined by the locality of the crossover operator.

Locality of GE Mutation and Duplication. We will now focus on the GE mutation and duplication operator. Both operators have high locality if $d(x^o, x^p)$ is low. We chose the same experimental setting as in our GE crossover study and applied either mutation ($p_m=0.01$) or duplication ($p_d = 1$) to all offspring that were generated during the random walks using all GE search operators. Just as crossover alone cannot increase the length of GE individuals, mutation alone cannot either; only duplication can increase the amount of genetic material (but not the diversity of the material).

Figure 4 plots the number of offspring x^o (cumulative relative frequency) over the distance $d(x^o, x^p)$ between offspring and corresponding parent. We omitted all cases where $d(x^o, x^p) = 0$ and plotted the results for $d(x^o, x^p) \leq 10$. Since p_m is low and many mutations and duplications have no effect on the encoded expression, many offspring are identical to their parents. For mutation, 98.8% of the 10 million offspring expressions are identical to their parents ($d(x^o, x^p)=0$). Only about 30% of the remaining offspring have a distance of 1 to their parents ($d(x^o, x^p) = 1$). For duplication, 53.3% of all offspring are identical to their parents. About 50% of the remaining offspring have $d(x^o, x^p) = 1$. Both local search operators suffer from low locality since they create offspring whose distances to their parents are large.

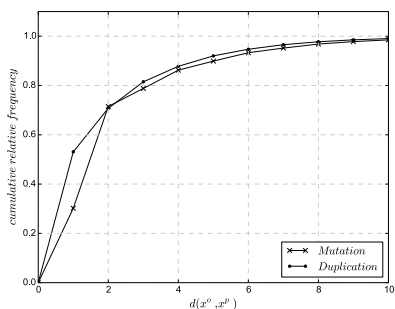


Fig. 4. GE with only mutation or duplication: number of offspring x^o (cumulative frequency) over $d(x^o, x^p)$

4 Conclusions

This work studies the locality of standard search operators for GE (crossover, mutation, and duplication) and GP (crossover and reproduction) by performing random walks through the search space of binary trees and measuring the distances between offspring and parents. The locality of standard search operators is high if the distances between the offspring and their parents is less than or equal to the distance between both parents. This concept is also known as the geometry of search operators. For binary trees we found out, that both GE and GP standard search operators have problems with low locality since a substantial number of offspring are not similar to their parents. Comparing GE and GP reveals that standard GE operators have higher locality than standard GP operators. The locality of the standard search operators in GE is mainly determined

by the crossover operator; mutation and duplication are less important. They are necessary to obtain a high diversity within the genetic material, but have low impact on the overall locality of the GE variation operators.

In the future we will extend this analysis to non-binary trees with more complex terminal and function sets. Although the results of the current study only hold for binary trees, we expect to see similar results for other tree structures. Moreover, we are going to use other distance metrics to measure similarities between individuals.

References

1. Byrne, J., O'Neill, M., McDermott, J., Brabazon, A.: An analysis of the behaviour of mutation in grammatical evolution. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 14–25. Springer, Heidelberg (2010)
2. Byrne, J., O'Neill, M., Brabazon, A.: Structural and nodal mutation in grammatical evolution. In: GECCO 2009: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1881–1882. ACM (2009)
3. Castle, T., Johnson, C.G.: Positional effect of crossover and mutation in grammatical evolution. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 26–37. Springer, Heidelberg (2010)
4. Doran, J., Michie, D.: Experiments with the graph traverser program. Proceedings of the Royal Society of London (A) 294, 235–259 (1966)
5. Galván-López, E., McDermott, J., O'Neill, M., Brabazon, A.: Towards an understanding of locality in genetic programming. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO 2010, pp. 901–908. ACM, New York (2010)
6. Galván-López, E., McDermott, J., O'Neill, M., Brabazon, A.: Defining locality as a problem difficulty measure in genetic programming. Genetic Programming and Evolvable Machines 12(4), 365–401 (2011)
7. Galvan-Lopez, E., O'Neill, M., Brabazon, A.: Towards understanding the effects of locality in gp. In: Eighth Mexican International Conference on Artificial Intelligence, MICAI 2009, pp. 9–14 (2009)
8. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3(5), 493–530 (1989)
9. Hugosson, J., Hemberg, E., Brabazon, A., O'Neill, M.: An investigation of the mutation operator using different representations in grammatical evolution. In: 2nd International Symposium “Advances in Artificial Intelligence and Applications”, Wisla, Poland, October 15–17, vol. 2, pp. 409–419 (2007)
10. Koza, J.R.: Genetic programming: On the programming of computers by natural selection. MIT Press, Cambridge (1992)
11. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine human-competitive machine intelligence. Springer, New York (2005)
12. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10(8), 707–710 (1966); Doklady Akademii Nauk SSSR 163(4), 845–848 (1965)
13. Liepkins, G.E., Vose, M.D.: Representational issues in genetic optimization. Journal of Experimental and Theoretical Artificial Intelligence 2, 101–115 (1990)

14. Moraglio, A.: Towards a Geometric Unification of Evolutionary Algorithms. PhD thesis, Department of Computer Science, University of Essex (November 2007)
15. Moraglio, A., Poli, R.: Topological interpretation of crossover. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1377–1388. Springer, Heidelberg (2004)
16. O’Neill, M., Ryan, C.: Grammatical evolution. *IEEE Transactions on Evolutionary Computation* 5(4), 349–358 (2001)
17. Radcliffe, N.J.: Equivalence class analysis of genetic algorithms. *Complex Systems* 5(2), 183–205 (1991)
18. Rothlauf, F.: *Distributed Autonomous Robotics Systems*, 1st edn. STUDEFUZZ, vol. 104. Springer, Heidelberg (2002)
19. Rothlauf, F.: *Design of Modern Heuristics*. Springer, Heidelberg (2011)
20. Rothlauf, F., Oetzel, M.: On the locality of grammatical evolution. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 320–330. Springer, Heidelberg (2006)
21. Ryan, C., Collins, J.J., Neill, M.O.: Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) EuroGP 1998. LNCS, vol. 1391, pp. 83–95. Springer, Heidelberg (1998)
22. Surry, P.D., Radcliffe, N.: Formal algorithms + formal representations = search strategies. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 366–375. Springer, Heidelberg (1996)
23. Uy, N.Q., Hoai, N.X., O’Neill, M., McKay, B.: The role of syntactic and semantic locality of crossover in genetic programming. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 533–542. Springer, Heidelberg (2010)
24. Uy, N.Q., O’Neill, M., Hoai, N.X., McKay, B., Galván-López, E.: Semantic similarity based crossover in gp: The case for real-valued function regression. In: Collet, P., Monmarché, N., Legrand, P., Schoenauer, M., Lutton, E. (eds.) EA 2009. LNCS, vol. 5975, pp. 170–181. Springer, Heidelberg (2010)
25. Uy, N.Q., Hoai, N.X., O’Neill, M., McKay, R.I., Phong, D.N.: On the roles of semantic locality of crossover in genetic programming. *Information Sciences* 235, 195–213 (2013)

Recurrent Cartesian Genetic Programming

Andrew James Turner¹ and Julian Francis Miller²

¹ The University of York, UK

`andrew.turner@york.ac.uk`

² The University of York, UK

`julian.miller@york.ac.uk`

Abstract. This paper formally introduces Recurrent Cartesian Genetic Programming (RCGP), an extension to Cartesian Genetic Programming (CGP) which allows recurrent connections. The presence of recurrent connections enables RCGP to be successfully applied to partially observable tasks. It is found that RCGP significantly outperforms CGP on two partially observable tasks: artificial ant and sunspot prediction. The paper also introduces a new parameter, *recurrent connection probability*, which biases the number of recurrent connections created via mutation. Suitable choices of this parameter significantly improve the effectiveness of RCGP.

1 Introduction

Cartesian Genetic Programming (CGP) [1] is a form of Genetic Programming (GP) [2] which encodes graph-based computational structures. CGP typically evolves acyclic programs which are only suited to fully observable tasks; when the desired outputs are purely a function of the current inputs. However, many tasks are partially observable and require that previous, as well as current, inputs be considered when calculating outputs. To be applicable to partially observable tasks CGP requires the ability to create programs which hold internal state information; that is to say, some form of memory/feedback. Previously traditional GP has been implemented with memory and feedback using explicit indexed memory [3] and Jordan type architectures [4] respectively.

This paper formally introduces Recurrent Cartesian Genetic Programming (RCGP), an extension to CGP which allows the creation of recurrent / cyclic graphs. RCGP has the ability, through feedback, to store internal state information making it suited to partially observable tasks. Recurrent connections are controlled by a new parameter, *recurrent connection probability*, which defines the likelihood of mutations creating a recurrent connection.

The aim of the paper is to apply and compare CGP and RCGP on partially observable tasks. The study has been undertaken to highlight that there are types of problems for which CGP is currently unsuitable, but to which RCGP can be successfully applied. The aim is not to compare RCGP's performance with other methods suited to partial observable tasks. This is left for further research.

The remainder of the paper is organized as follows: Section 2 describes CGP, Section 3 introduces RCGP, Section 4 describes the experiments used to compare CGP and RCGP, Section 5 describes the benchmarks used in the experiments, Section 6 presents the results, Section 7 gives a discussion of the findings and finally Section 8 gives closing conclusions.

2 Cartesian Genetic Programming

CGP [1] [5] is a form of GP [2] which typically evolves acyclic computational structures of nodes (graphs) indexed by their Cartesian coordinates. CGP does not suffer from bloat [6] [7]; a drawback of many GP methods [8]. CGP chromosomes contain non-functioning genes enabling neutral genetic drift during evolution [9] [10]. CGP typically uses point or probabilistic mutation, no crossover and a $(1 + \lambda)$ -ES. Although CGP chromosomes are of static size, the number of active nodes varies during evolution enabling variable length phenotypes. The user therefore specifies a *maximum* number nodes, of which only a proportion will be active. Overestimating the number of nodes has shown to greatly aid evolution [11]; which is thought to heighten neutral genetic drift but could also be compensating for length bias [12].

Each CGP chromosome comprises of function genes (F_i), connection genes (C_i) and output genes (O_i). The function genes represent indexes in a function look-up-table and describe the functionality of each node. The connection genes describe from where each node gathers its inputs. For regular acyclic CGP, connection genes may connect a given node to any previous node in the program, or any of the program inputs. The output genes address any program input or internal node and define which are used as program outputs.

Originally CGP programs were organized with nodes arranged in rows (nodes per layer) and columns (layers); with each node indexed by its row and a column. However, this is an unnecessary constraint, as any configuration possible using a given number of rows and columns is also possible using one row with many columns; provided the total number of nodes remains constant. This is due to CGP being capable of evolving where each node connects its inputs. Consequently, here the chromosomes are defined with one row and n columns; with each node only indexed by its column. A generic (one row) CGP chromosome is given in Equation 1; where α is the arity of each node, n is the number of nodes and m is the number of program outputs.

$$F_0 C_{0,0} \dots C_{0,\alpha} F_1 C_{1,0} \dots C_{1,\alpha} \dots F_n C_{n,0} \dots C_{n,\alpha} O_0 \dots O_m \tag{1}$$

An example CGP program is given in Figure 1 along with its corresponding chromosome. As can be seen, all nodes are connected to previous nodes or program inputs. Not all program inputs have to be used, enabling evolution to decide which inputs are significant. An advantage of CGP over tree-based GP, again seen in Figure 1, is that node outputs can be reused multiple times, rather than requiring the same value to be recalculated if it is needed again. Finally, not all nodes contribute to the final program output, these represent the inactive nodes which enable neutral genetic drift and make variable length phenotypes possible.

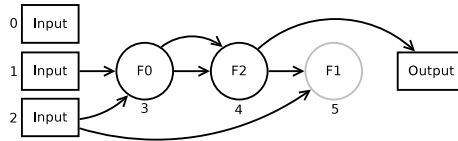


Fig. 1. Example CGP program corresponding to the chromosome: 012 233 124 4

3 Recurrent Cartesian Genetic Programming

Although RCGP has never been formally presented, it has been previously discussed as a possible extension to CGP [1]. RCGP has also been used as a method for removing length bias¹ [12] when investigating why CGP does not suffer from bloat [7]. Additionally a form of CGP has been used which implemented a Jordan type architecture [13] for allowing feedback [14]. Here the application was Cartesian Genetic Programming of Artificial Neural Networks (CGPANN) [15,16]; a NeuroEvolutionary technique based on CGP. Although using Jordan type architectures represents a simple method for allowing recurrent connections, it does so in a very restricted form. For instance, the user must decide in advance how many and what type of recurrent connections will be used.

3.1 Recurrent Cartesian Genetic Programming Implementation

In CGP, connection gene values are restricted so as to only allow acyclic connections. In RCGP this restriction is lifted so as to allow connections between a given node and *any other* node in the program (including itself) or program inputs. An example program which could be generated using RCGP is given in Figure 2 along with the corresponding chromosome.

RCGP phenotypes are executed similar to CGP phenotypes. Starting at the active node closest to the inputs, each node calculates its output value based on its inputs. Once all active nodes have been updated, the program outputs are recorded. However, with the presence of recurrent connections, a nodes output can be required before it has been calculated. To deal with this, all the nodes are initialised to output zero until they calculate their own value. Therefore when executing a RCGP phenotype the the following process is used:

1. set all active nodes to output zero
2. apply the next set of program inputs
3. update all active nodes **once** from program inputs to program outputs
4. read the program outputs
5. repeat from 2 until all program input sets have been applied

It should be noted that the program outputs are read *once* for each set of applied program inputs. It would also be possible to execute the program multiple

¹ Although through email correspondence with Brian Goldman it may be the case that RCGP only serves to alter the length bias rather than remove it.

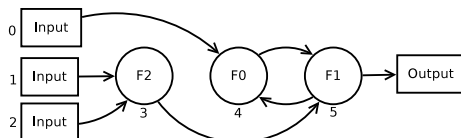


Fig. 2. Example RCGP program cosponsoring to the chromosome: 212 005 134 5

times for each set of program inputs. In such a case, the average of the program outputs or the settled program outputs² could be taken. The method described here was chosen for its simplicity and because there is no guarantee that the program outputs would ever settle.

A drawback of placing no constraints on connection gene values is that, on average, mutations to connection genes will result in as many feed-forward connections as recurrent. As it is highly unlikely that many tasks will require fifty percent of connections to be recurrent, this places a bias towards possibly unsuitable areas in the solution space. For this reason, a new parameter is introduced which controls the likelihood of mutations creating recurrent connections. This parameter is called *recurrent connection probability*. A recurrent connection probability of zero percent results in only feed-forward connections (i.e. regular CGP). A recurrent connection probability of fifty percent results in mutations causing as many feed-forward connections as recurrent (i.e. RCGP without the new parameter). A recurrent connection probability of one hundred percent results in only recurrent connections. It should be noted that this parameter does not directly control the number of recurrent connections, only the probability of mutations creating recurrent connections.

An important property of CGP is that the active nodes can be determined before executing the program. This is significant as a high proportion of nodes are often inactive [11] and calculating their outputs wastes computation time. To determine which nodes are active the following algorithm is used [1]: 1) add each program output node to a list of active nodes 2) for each node added to the active node list, add the nodes to which they also connect 3) continue until the program inputs are reached. Determining the active nodes for RCGP follows a similar algorithm except only nodes which are not currently in the active node list are added. This extra criteria breaks cycles enabling active nodes to be determined for RCGP.

3.2 Implications of Recurrent Connections

An implication of RCGP is that it is now possible for chromosomes to describe phenotypes where none of the active nodes connect to the program inputs. These programs are therefore unsuited to any realistic task. However such programs will likely score a low fitness and be quickly dropped from the population.

² Where settled output refers to the converged program output value(s) after many updates of the active nodes whilst applying the same program inputs.

Another implication of allowing recurrent connections occurs when applying RCGP to tasks where each set of inputs are independent from each other. For example, suppose we are trying to evolve a program that can implement a six bit parity circuit. Normally, we think of each line of the truth table as being independent of one another (i.e. the order in which each set of inputs occurs is unimportant). If the fitness function always tests each line of the truth table in the same order, RCGP could in principle use previous inputs to “predict” the correct output. This was shown to be the case in [7]. In an extreme case, RCGP could “predict” the correct outputs without ever considering the program inputs³. It is therefore important that RCGP should only be applied to tasks where the series of inputs are related, such as in time series prediction; otherwise additional precautions would be required to prevent this behaviour.

4 Experiments

The experiments presented are designed to test if RCGP is a suitable extension to CGP when solving partially observable tasks. As RCGP is implemented using the recurrent connection probability, this parameter is varied over [0, 10, 20, 50] percent. Where zero percent is equivalent to CGP, fifty percent is equivalent to RCGP without the additional parameter and ten and twenty percent represents RCGP with lower biases for recurrent connections.

If RCGP achieves statistically significantly better fitness than CGP on the given tasks, then RCGP will be considered a suitable extension to CGP when solving partially observable problems. If biasing the level of recurrence is shown to statistically significantly influence fitness, then the recurrent connection probability will be considered a suitable parameter for RCGP.

As this is the first time RCGP has been investigated it is unknown how the number of available nodes will influence results. For this reason each experiment is repeated over a range of available nodes [10, 20, ..., 90, 100] to ensure a fair comparison between CGP and RCGP. Other than the parameters previously given, the following are used throughout the experiments: (1 + 4)-ES, 3% probabilistic mutation and a node arity of two. The results presented are the average fitness of fifty independent runs. Each run is given ten thousand generations before terminating the search.

5 Benchmarks

Two partially observable benchmarks are used in the described experiments, Artificial Ant and Sunspots. The Artificial Ant benchmark is a reinforcement learning control task and the Sunspots benchmark is a supervised learning series forecasting task.

³ This was shown to be the case in unrepresented results where RCGP “solved” the six bit parity task without any inputs!

5.1 Artificial Ant

The Artificial Ant problem [17] is a classic challenging [18] benchmark commonly used by GP [2]. The task is to design a controller which navigates an ant around a toroidal map maximising food intake. The ant can only perceive whether the location ahead of its current position contains food. Each time step the ant undertakes one of four actions: move forward, turn left 90°, turn right 90° or do nothing. The map used here is the “Santa Fe Ant Trail” [2] given in Figure 3.

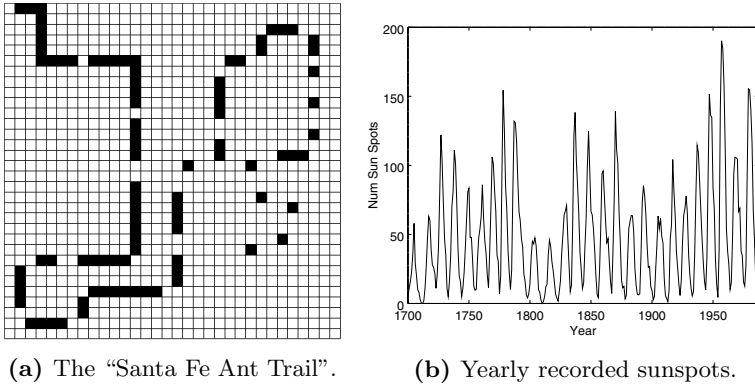


Fig. 3. (a) Depiction of the “Santa Fe Ant Trail”. Black and white represent food and no food respectively. (b) Yearly number of recorded sunspots.

In this paper the form of the controller differs from that commonly used by GP [2]. Here the evolved program’s inputs describe if the location ahead contains food and the program’s outputs are decoded into one of the possible four actions; this is not dissimilar to the original implementation [17]. Other GP implementations [2] create programs where the program inputs are the possible actions and the program outputs are unused. The function set used by the nodes causes the inputs (actions) to either be implemented outright or to be conditional on whether food is ahead. Once the program outputs are reached the program starts over. CGP has previously been applied to the benchmark in its more commonly used form [5].

In this paper, the evolved controllers have two mutually exclusive inputs, whereby the first input is set as ‘1’ if the location ahead of the ant contains food, else it is set as ‘0’. The controller has two outputs, where: [1 1] represents move forward, [0 1] turn right, [1 0] turn left and [0 0] do nothing. The ant starts in the top left (0,0) of the toroidal map facing east and is allowed 400 time steps to consume as much food as possible. The amount of food eaten is then used as the fitness measure; out of a maximum 89. The function set used comprises of the four Boolean logic gates: AND, OR, NOT, and XOR.

5.2 Sunspots

The Sunspots benchmark [19] is a commonly used [20] time series prediction benchmark which describes the number of observed sunspots dating back to 1700. The data was recorded by the SIDC-team, at the World Data Center for the Sunspot Index, Royal Observatory of Belgium [19]. The dataset contains the yearly number of recorded sunspots between 1700 and 1987; given in Figure 3. The first 221 years (1700-1920) are used as the training set with the remaining 67 years (1921-1987) used as the testing set.

Most series forecasters which are applied to the Sunspots benchmark use multiple inputs consisting of the current and previous years number of sunspots. However, in this paper only one input is used which gives the current number of sunspots. This restriction to one input was imposed to force the task to become partially observable. This restriction also makes the task much more challenging since any trends in the data must be calculated internally as the data is passed in year by year. The single output is the predicted number of sunspots 35 years ahead of the current input. The single input to the series forecaster is normalised into a $[0, 1]$ range by dividing by two hundred (a value greater than the highest number of sunspots in any observed year). The single output is also multiplied by two hundred before being used as the predicted number of sunspots.

The fitness measure is the mean average error (MAE) given by: $\frac{1}{N} \sum_{i=1}^N |e_i|$ where N is the number of samples and e is the difference between the actual and predicted number of sunspots. The function set used for this task comprises of ten symbolic expressions: $x_1 + x_2$, $x_1 - x_2$, $x_1 \times x_2$, $x_i \div x_j$, $|x_1|$, x_1^2 , x_1^3 , e^{x_1} , $\sin(x_1)$ and $\cos(x_1)$. Where x_1 and x_2 are the two inputs to each node and the division operator is protected so as to return one when dividing by zero.

6 Results

The the average fitnesses (from 50 runs) achieved using RCGP are given for the Artificial Ant and Sunspots benchmarks in Figure 4. It should be recalled that a recurrent connection probability of zero percent is equivalent to regular CGP. The average generalisation performance on the Sunspot testing set is also given in Figure 5 along with an example forecaster created using one hundred nodes and a recurrent connection probability of ten percent.

To identify if the differences due to the recurrent connection probability seen in Figure 4 are statistically significant the results are analysed using the non-parametric two sided Mann-Whitney U-test. When using the U-test $p < 0.05$ indicates statistical significance between two sets of data. Tables 1 and 2 give the p values when comparing pairs of recurrent connection probabilities using ten, twenty and one hundred nodes for the Artificial Ant and Sunspot (training set) benchmarks.

As can be seen in Figure 4 and Tables 1 and 2, a recurrent connection probability of zero percent consistently performs worst on both benchmarks with statistical significance. This demonstrates that there are types of tasks which regular CGP is not suited to but to which RCGP can be successfully applied.

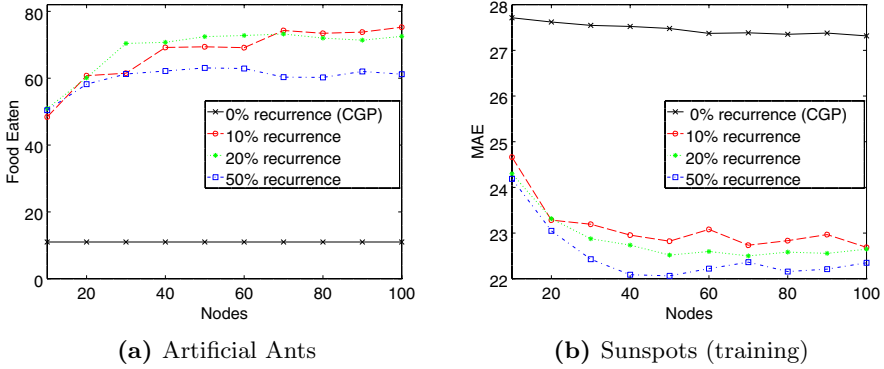


Fig. 4. Results of applying RCGP on the two benchmark

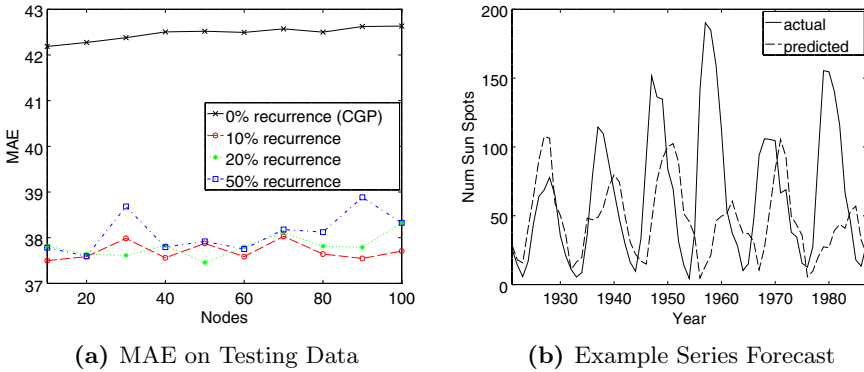


Fig. 5. Generalisation of RCGP on the Sunspot test set

When comparing recurrent connection probabilities, it can be seen that lower percentages (ten and twenty) give statistically significantly better results on the Artificial Ant benchmark than higher percentages (fifty). On the Sunspots benchmark the opposite is true, with higher levels producing the best results on the training set. However, this greater performance on the training set is accompanied by weaker generalisation on the testing set; seen in Figure 5.

7 Discussion

The results given in Section 6 clearly demonstrate that CGP is unsuitable for partially observable tasks. This is not a surprising result as CGP has no capacity to recall previous inputs or infer internal state information. For instance the best strategy CGP could find for the Artificial Ant task was to rotate until food is ahead, and then move forward.

Table 1. Artificial Ant: p values comparing pairs of recurrent connection probabilities

	0%	10%	20%	50%		0%	10%	20%	50%		0%	10%	20%	50%
0%	1	~ 0	~ 0	~ 0	0%	1	~ 0	~ 0	~ 0	0%	1	~ 0	~ 0	~ 0
10%	-	1	0.274	0.118	10%	-	1	0.391	0.025	10%	-	1	0.260	2E-5
20%	-	-	1	0.576	20%	-	-	1	0.002	20%	-	-	1	2E-4
50%	-	-	-	1	50%	-	-	-	1	50%	-	-	-	1

(a) 10 Nodes (b) 50 Nodes (c) 100 Nodes

Table 2. Sunspots: p values comparing pairs of recurrent connection probabilities

	0%	10%	20%	50%		0%	10%	20%	50%		0%	10%	20%	50%
0%	1	~ 0	~ 0	~ 0	0%	1	~ 0	~ 0	~ 0	0%	1	~ 0	~ 0	~ 0
10%	-	1	0.022	0.021	10%	-	1	0.234	0.011	10%	-	1	0.671	0.312
20%	-	-	1	0.759	20%	-	-	1	0.167	20%	-	-	1	0.458
50%	-	-	-	1	50%	-	-	-	1	50%	-	-	-	1

(a) 10 Nodes (b) 50 Nodes (c) 100 Nodes

The results in Section 6 show that RCGP *is* highly suited for partially observable tasks. For both benchmarks it dramatically and statistically significantly outperforms CGP.

Section 6 also showed that simply allowing mutation to create feed-forward or recurrent connections with equal probability does not always produce the best results. This is because it is unlikely that a given task will require fifty percent of connections to be recurrent. The introduction of the recurrent connection probability parameter allows the user to bias mutations so as to produce greater or fewer recurrent connections. It has been shown that using a recurrent connection probability of fifty percent (i.e. effectively not using the recurrent connection probability parameter) produces poor results on the Artificial Ant benchmark and causes over training on the Sunspot benchmark⁴. Using a recurrent connection probability of ten percent produced the best results on the Artificial Ant benchmark and produce the best MAE on the testing set for the Sunspot benchmark. The recurrent connection probability is therefore an important additional parameter when using RCGP.

8 Conclusion

RCGP is an extension to CGP which enables application to partially observable tasks. On two partially observable benchmark problems, Artificial Ant and Sunspot prediction, RCGP gives statistically significant improvements compared with acyclic CGP. RCGP has been implemented using a recurrent connection probability parameter which biases the number of recurrent connections created

⁴ Over training could be controlled via the use of a validation set but this was not considered here.

via mutations. This is introduced as simply allowing mutations to connect any two nodes creates an unhelpful bias for as many feed-forward connections as recurrent. Further research is needed to compare the performance of RCGP with other methods suited to partially observable problems and to apply RCGP to additional domains; such as creating recurrent artificial neural networks.

References

1. Miller, J.F. (ed.): Cartesian Genetic Programming. Springer (2011)
2. Koza, J.R.: Genetic Programming: On the programming of computers by means of natural selection, vol. 1. MIT Press (1992)
3. Teller, A.: Turing completeness in the language of genetic programming with indexed memory. In: IEEE Evolutionary Computation, pp. 136–141 (1994)
4. Teredesai, A., Govindaraju, V., Ratzlaff, E., Subrahmonia, J.: Recurrent genetic programming. In: 2002 IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 5–9. IEEE (2002)
5. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
6. Miller, J.F.: What bloat? Cartesian genetic programming on Boolean problems. In: Genetic and Evolutionary Computation Conference, pp. 295–302 (2001)
7. Turner, A.J., Miller, J.F.: Cartesian Genetic Programming: Why No Bloat? In: Genetic Programming: 17th European Conference (to appear, 2014)
8. Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines 10(2), 141–179 (2009)
9. Vassilev, V.K., Miller, J.F.: The Advantages of Landscape Neutrality in Digital Circuit Evolution. In: Miller, J.F., Thompson, A., Thompson, P., Fogarty, T.C. (eds.) ICES 2000. LNCS, vol. 1801, pp. 252–263. Springer, Heidelberg (2000)
10. Yu, T., Miller, J.F.: Neutrality and the evolvability of boolean function landscape. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tetamanzi, A.G.B., Langdon, W.B. (eds.) EuroGP 2001. LNCS, vol. 2038, pp. 204–217. Springer, Heidelberg (2001)
11. Miller, J.F., Smith, S.: Redundancy and computational efficiency in Cartesian genetic programming. Evolutionary Computation 10(2), 167–174 (2006)
12. Goldman, B.W., Punch, W.F.: Length bias and search limitations in Cartesian genetic programming. In: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, pp. 933–940. ACM (2013)
13. Jordan, M.I.: Serial order: A parallel distributed processing approach. Technical report, Institute for Cognitive Science (1986)
14. Khan, M., Khan, G., Miller, J.: Efficient representation of recurrent neural networks for markovian/non-markovian non-linear control problems. In: IEEE Intelligent Systems Design and Applications, pp. 615–620 (2010)
15. Khan, M.M., Ahmad, M.A., Khan, M.G., Miller, J.F.: Fast learning neural networks using Cartesian Genetic Programming. Neurocomputing 121, 274–289 (2013)
16. Turner, A.J., Miller, J.F.: Cartesian Genetic Programming encoded Artificial Neural Networks: A Comparison using Three Benchmarks. In: Genetic and Evolutionary Computation, pp. 1005–1012 (2013)

17. Jefferson, D., Collins, R., Cooper, C., Dyer, M., Flowers, M., Korf, R., Taylor, C., Wang, A.: The genesys system: Evolution as a theme in artificial life. In: *Artificial Life*. Addison-Wesley, Redwood City (1990)
18. Langdon, W.B., Poli, R.: Why ants are hard. Technical report, School of Computer Science, The University of Birmingham, Birmingham, UK (1998)
19. SIDC-team: The International Sunspot Number. Monthly Report on the International Sunspot Number, online catalogue (1700-1987)
20. Khashei, M., Bijari, M.: An artificial neural network (p,d,q) model for timeseries forecasting. *Expert Systems with Applications* 37(1), 479–489 (2010)

An Analysis on Selection for High-Resolution Approximations in Many-Objective Optimization

Hernán Aguirre¹, Arnaud Liefooghe², Sébastien Verel³, and Kiyoshi Tanaka¹

¹ Faculty of Engineering, Shinshu University
4-17-1 Wakasato, Nagano, 380-8553 Japan

² Université Lille 1 LIFL, UMR CNRS 8022,
Inria Lille-Nord Europe, France

³ Université du Littoral Côte d'Opale, LISIC, 62228 Calais, France
{ahernan,ktanaka}@shinshu-u.ac.jp, arnaud.liefooghe@lifl.fr,
verel@univ-littoral.fr

Abstract. This work studies the behavior of three elitist multi- and many-objective evolutionary algorithms generating a high-resolution approximation of the Pareto optimal set. Several search-assessment indicators are defined to trace the dynamics of survival selection and measure the ability to simultaneously keep optimal solutions and discover new ones under different population sizes, set as a fraction of the size of the Pareto optimal set.

1 Introduction

In multi-objective optimization the aim of the optimizer is to find a good approximation of the Pareto optimal set (POS) in terms of convergence and diversity of solutions. Convergence dictates that solutions in the approximation must be either members of the POS or close to it in objective space. Diversity usually implies that solutions in the approximation should be evenly spaced in objective space, following the distribution of the POS.

In many-objective optimization, in addition to convergence and diversity, a third criterion also becomes a relevant aim of the optimizer. We call it the *resolution* of the approximation. The *resolution* is related to the number of points in the generated approximation of the POS. In many-objective problems the number of solutions in the POS increases exponentially [1] with the dimensionality of the objective space. In general, many more points are required to cover uniformly with the same density a higher dimensional space. However, the required resolution of the approximation could vary depending on the application domain and the task of the optimization within the problem solving approach. A low resolution of the approximation may suffice in some domain applications. For example, domains where the formulation of the problem is already well understood and a solution has to be found and implemented regularly, such as the daily operational schedule of machines and the jobs assigned to them in a manufacturing plant. In these domains, the optimizer is often required to provide

alternative exact solutions and too many of them could overwhelm a decision maker (operations manager) that must suggest a prompt course of action. In other application domains a high resolution of the approximation is required. For example in design optimization, where the problem-solving cycle often starts with multiple, sometimes ill defined, problem formulations and uses optimization as a tool to validate the understanding of the problem and to discover new features about it. In these application domains it is not unusual to require that the optimizer provides approximations of the POS with tens of thousands or even hundreds of thousands of solutions. These approximations are subjected to data mining and analysis to verify and improve the problem formulation itself, understand the tradeoffs between variables and objectives, and extract valuable design knowledge [2]. Thus, a many-objective optimizer should also aim to find an approximation with a *resolution* that properly captures the POS, with enough points to provide a useful description of it, depending on the dimensionality of the objective space and the optimization task at hand.

Many-objective optimization was initially attempted using evolutionary algorithms that proved effective for two and three objectives only to discover their lack of scalability. A significant part of the research effort has been understanding the reasons for their failure and improving them, particularly in terms of convergence. Recently, some many-objective optimizers are being proposed [3,4]. However, the performance of the improved and newly proposed algorithms is commonly assessed using a relatively very small number of points focusing mostly on convergence and/or diversity. The resolution of the approximation in many-objective optimization has not been deeply studied and it is not clear the capabilities and behavior of the algorithms under this additional important criterion.

In this work we analyze the behavior of three elitist multi- and many-objective evolutionary algorithms generating a high-resolution approximation of the POS. We define a basic indicator for resolution, the accumulated gain of the population, and several generational search-assessment indices respect to the POS. We trace the dynamics of survival selection and study the ability to simultaneously keep Pareto optimal (PO) solutions in the population and find new ones to improve the resolution of the approximation, setting population size as a fraction of the size of the POS. We use MNK-landscapes with 3 – 6 objectives and 20 bits, for which it is possible to know by enumeration all PO solutions.

2 Methodology

An important objective of this work is to analyze the ability of the algorithms to generate a high-resolution approximation of the POS. A simple and basic indicator for resolution is to count the number of PO solutions found by the algorithms. In many-objective problems is likely that the population size is considerable smaller than the size of the POS. Thus, to achieve a good resolution the algorithms should first be able to hit the POS with some members of the population and then continue discovering other PO solutions. The ability to

Table 1. Number of Pareto optimal solutions $|POS|$ and number of non-dominated *Fronts* in the landscapes with $M = 3, 4, 5,$ and 6 objectives. Also, fraction of $|P| / |POS|$ (in %) for various population sizes $|P|$ investigated in this study.

M	$ POS $	$Fronts$	$ P / POS $ (%)								
			50	100	200	500	1,000	2,000	4,000	5,600	11,200
3	152	258	32.9	65.8	132.6						
4	1,554	76	3.2	6.4	12.9	32.2	64.4				
5	6,265	29	0.8	1.6	3.2			31.9	63.8		
6	16,845	22	0.3	0.6	1.2					33.2	66.5

converge towards the POS is a very important feature of the algorithm. In this study, we focus our attention mostly on the ability of the algorithm to continue discovering new PO solutions assuming that the algorithms can converge to the POS.

To evaluate this ability we use four MNK-landscapes [1] randomly generated with $M = 3, 4, 5, 6$ objectives, $N = 20$ bits, and $K = 1$ epistatic bit. In small landscapes with low non-linearity it is relatively simple for the algorithm to hit the optimal set. It is also possible to enumerate them and know the POS in order to analyze the dynamic of the algorithms respect to the optimum set. The exact number of PO solutions found by enumeration and the number of non-dominated fronts are shown in Table 1 under columns $|POS|$ and *Fronts*, respectively. The same table also shows the corresponding fraction (%) of the population sizes $|P|$ to the $|POS|$ for various population sizes investigated here.

We run the algorithms for a fixed number of T generations, collecting in separate files the sets of non-dominated solutions $\mathcal{F}_1(t)$ found at each generation. The approximation of the POS for a run of the algorithm, denoted $\mathcal{A}(T)$, is built by computing the non-dominated set from all generational non-dominated sets $\mathcal{F}_1(t), t = 0, 1, \dots, T$, making sure no duplicate solutions are included. In general, the approximation at generation t is given by

$$\mathcal{A}(t) = \{ \mathbf{x} : \mathbf{x} \in \mathcal{X}(t) = \mathcal{A}(t-1) \cup \mathcal{F}_1(t) \setminus \mathcal{A}(t-1) \cap \mathcal{F}_1(t) \wedge \nexists \mathbf{y} \in \mathcal{X}(t) \mathbf{y} \succeq \mathbf{x} \} \quad (1)$$

$$\mathcal{A}(0) = \mathcal{F}_1(0), \quad (2)$$

where $\mathbf{y} \succeq \mathbf{x}$ denotes solution \mathbf{y} Pareto dominates solution \mathbf{x} . The basic resolution index α of the approximation at generation t is,

$$\alpha(t) = \frac{|\{ \mathbf{x} : \mathbf{x} \in \mathcal{A}(t) \wedge \mathbf{x} \in POS \}|}{|POS|}, \quad (3)$$

which gives the fraction of the accumulated number of PO solutions found until generation t to the size of the POS. The highest resolution of the generated approximation of the POS is achieved when all Pareto optimal solutions are found.

Table 2. Generational search-assessment indices I_t . Measures are taken on non-dominated population $\mathcal{F}_1(t)$ with respect to $\mathcal{F}_1(t-1)$ and/or the POS, normalized by population size $|P|$.

I_t	Formula	Comment
τ_t	$ \{\mathbf{x} : \mathbf{x} \in \mathcal{F}_1(t) \wedge \mathbf{x} \in POS\} / P $	PO solutions
τ_t^-	$ \{\mathbf{x} : \mathbf{x} \in \mathcal{F}_1(t) \wedge \mathbf{x} \in \mathcal{F}_1(t-1) \wedge \mathbf{x} \in POS\} / P $	Old PO solutions
τ_t^+	$ \{\mathbf{x} : \mathbf{x} \in \mathcal{F}_1(t) \wedge \mathbf{x} \notin \mathcal{F}_1(t-1) \wedge \mathbf{x} \in POS\} / P $	Possibly new PO solutions
τ_t^*	$ \{\mathbf{x} : \mathbf{x} \in \mathcal{F}_1(t) \wedge \mathbf{x} \notin \cup_{k=1}^{t-1} \mathcal{F}_1(k) \wedge \mathbf{x} \in POS\} / P $	Absolutely new PO solutions
δ_t	$ \{\mathbf{x} : \mathbf{x} \in \mathcal{F}_1(t-1) \wedge \mathbf{x} \notin \mathcal{F}_1(t) \wedge \mathbf{x} \in POS\} / P $	Dropped PO solutions
γ_t	$ \{\mathbf{x} : \mathbf{x} \in \mathcal{F}_1(t) \wedge \mathbf{x} \notin POS\} / P $	Non-dominated, not PO sol.

Similarly, the accumulated population gain at generation t can be expressed as

$$\beta(t) = \frac{|\{\mathbf{x} : \mathbf{x} \in \mathcal{A}(t) \wedge \mathbf{x} \in POS\}|}{|P|}. \quad (4)$$

For our analysis on the dynamics of the algorithm, we compare the sets $\mathcal{F}_1(t)$ with the POS to determine which solutions in $\mathcal{F}_1(t)$ are Pareto optimal and compute several generational search-assessment indices I_t (τ_t , τ_t^* , τ_t^+ , τ_t^- , δ_t , γ_t), as shown in Table 2. Note that these generational indexes are expressed as a fraction of the population size $|P|$. In this work we analyze them and their average value \bar{I} ($\bar{\tau}$, $\bar{\tau}^*$, $\bar{\tau}^+$, $\bar{\tau}^-$, $\bar{\delta}$, $\bar{\gamma}$) taken over all generations computed as $\bar{I} = \frac{1}{T+1} \sum_{t=0}^T I_t$.

In this work we analyze NSGA-II [5], IBEA [6], and the Adaptive ε -Sampling and ε -Hood algorithm [4]. In the following we briefly describe these algorithms, particularly fitness assignment, survival selection, and parent selection.

3 Algorithms

3.1 NSGA-II

NSGA-II is an elitist algorithm that uses Pareto dominance and crowding estimation of solutions for survival and parent selections. To compute fitness of the individuals, the algorithm joins the current population \mathcal{P}_t with its offspring \mathcal{Q}_t and divide it in non-dominated fronts $\mathcal{F} = \{\mathcal{F}_i, i = 1, 2, \dots, N_F\}$ using the non-dominated sorting procedure. It also calculate the crowding distance d_j of solutions within the fronts \mathcal{F}_i . The fitness of j -th solution in the i -th front is $\text{Fitness}(\mathbf{x}_j) = (i, d_j)$, where the front number is the primary rank and crowding distance the secondary rank. Survival selection is performed by copying iteratively the sets of solutions \mathcal{F}_i to the new population \mathcal{P}_{t+1} until it is filled. If the set \mathcal{F}_i , $i > 1$, overfills the new population \mathcal{P}_{t+1} , the required number of solutions are chosen based on their secondary rank d_j . Parent selection for reproduction consists of binary tournaments between randomly chosen individuals from \mathcal{P}_{t+1} using their primary rank i to decide the winners, breaking ties with their secondary rank d_j .

3.2 IBEA (Indicator-Based Evolutionary Algorithm)

The main idea of IBEA [6] is to introduce a total order between solutions by means of an arbitrary binary quality indicator I . The fitness assignment scheme of IBEA is based on a pairwise comparison of solutions in a population with respect to indicator I . Each individual \mathbf{x} is assigned a fitness value measuring the “loss in quality” if \mathbf{x} was removed from the population P , i.e., $\text{Fitness}(\mathbf{x}) = \sum_{\mathbf{x}' \in P \setminus \{\mathbf{x}\}} (-e^{-I(\mathbf{x}', \mathbf{x})/\kappa})$, where $\kappa > 0$ is a user-defined scaling factor. Survival selection is based on an elitist strategy that combines the current population \mathcal{P}_t with its offspring \mathcal{Q}_t , iteratively deletes worst solutions until the required population size is reached, and assigns the resulting population to $\mathcal{P}_{(t+1)}$. Here, each time a solution is deleted the fitness values of the remaining individuals are updated. Parent selection for reproduction consists of binary tournaments between randomly chosen individuals using their fitness to decide the winners.

Different indicators can be used within IBEA. We here choose to use the binary additive ϵ -indicator ($I_{\epsilon+}$), as defined by the original authors [6].

$$I_{\epsilon+}(\mathbf{x}, \mathbf{x}') = \max_{i \in \{1, \dots, n\}} \{f_i(\mathbf{x}) - f_i(\mathbf{x}')\} \tag{5}$$

$I_{\epsilon+}(\mathbf{x}, \mathbf{x}')$ gives the minimum value by which a solution $\mathbf{x} \in P$ has to, or can be translated in the objective space in order to weakly dominate another solution $\mathbf{x}' \in P$. More information about IBEA can be found in [6].

3.3 The A ϵ S ϵ H

Adaptive ϵ -Sampling and ϵ -Hood (A ϵ S ϵ H) [4] is an elitist evolutionary many-objective algorithm that applies ϵ -dominance principles for survival selection and parent selection. There is not an explicit fitness assignment method in this algorithm.

Survival selection joins the current population \mathcal{P}_t and its offspring \mathcal{Q}_t and divide it in non-dominated fronts $\mathcal{F} = \{\mathcal{F}_i\}, i = 1, 2, \dots, N_F$ using the non-dominated sorting procedure. In the rare case where the number of non-dominated solutions is smaller than the population size $|\mathcal{F}_1| < |P|$, the sets of solutions \mathcal{F}_i are copied iteratively to \mathcal{P}_{t+1} until it is filled; if set $\mathcal{F}_i, i > 1$, overfills \mathcal{P}_{t+1} , the required number of solutions are chosen randomly from it. On the other hand, in the common case where $|\mathcal{F}_1| > |P|$, it calls ϵ -sampling with parameter ϵ_s . This procedure iteratively samples randomly a solution from the set \mathcal{F}_1 , inserting the sample in \mathcal{P}_{t+1} and eliminating from \mathcal{F}_1 solutions ϵ -dominated by the sample. After sampling, if \mathcal{P}_{t+1} is overfilled solutions are randomly eliminated from it. Otherwise, if there is still room in \mathcal{P}_{t+1} , the required number of solutions are randomly chosen from the initially ϵ -dominated solutions and added to \mathcal{P}_{t+1} .

After survival selection there is not an explicit ranking that could be used to bias mating. Rather, for parent selection the algorithm first uses a procedure called ϵ -hood creation to cluster solutions in objective space. This procedure randomly selects an individual from the surviving population and applies ϵ -dominance with parameter ϵ_h . A neighborhood is formed by the selected solution

and its ε_h -dominated solutions. Neighborhood creation is repeated until all solutions in the surviving population have been assigned to a neighborhood. Parent selection is implemented by the procedure ε -hood mating, which sees neighborhoods as elements of a list than can be visited one at the time in a round-robin schedule. The first two parents are selected randomly from the first neighborhood in the list. The next two parents will be selected randomly from the second neighborhood in the list, and so on. When the end of the list is reached, parent selection continues with the first neighborhood in the list. Thus, all individuals have the same probability of being selected within a specified neighborhood, but due to the round-robin schedule individuals belonging to neighborhoods with fewer members have more reproduction opportunities than those belonging to neighborhoods with more members.

Both epsilon parameters ε_s and ε_h used in survival selection and parent selection, respectively, are dynamically adapted during the run of the algorithm. Further details about A ε S ε H can be found in [4].

4 Experimental Results and Discussion

4.1 Operators of Variation and Parameters

In all algorithms we use two point crossover with rate $pc = 1.0$, and bit flip mutation with rate $pm = 1/n$. In A ε S ε H we set the reference neighborhood size H_{size}^{Ref} to 20 individuals. The mapping function $\mathbf{f}(\mathbf{x}) \mapsto^\varepsilon \mathbf{f}'(\mathbf{x})$ used for ε -dominance in ε -sampling truncation and ε -hood creation is additive, $f'_i = f_i + \varepsilon, i = 1, 2, \dots, m$. For IBEA, the scaling factor is set to $\kappa = 0.001$. The algorithms run for $T = 100$ generations. Results analyzed here were obtained from 30 independent runs of the algorithms.

4.2 Accumulated Number of Pareto Optimal Solutions Found

Fig.1 shows the the basic resolution index $\alpha(T)$ of the approximation at the end of the run, i.e. the ratio of accumulated number of PO solutions found to the size of the POS. Results are shown for 3, 4, 5, and 6 objectives using population sizes of $\{50, 100, 200\}$. Similarly, Fig.2 shows results for 5, and 6 objectives using larger population sizes, between 500 and 11,200 individuals.

Note that A ε S ε H finds many more Pareto optimal solutions than NSGA-II and IBEA for all population sizes and number of objectives tried here, whereas NSGA-II finds more solutions than IBEA when population sizes are relatively a large fraction of the size of the POS. See Fig.1 (a) and Fig.2 (a)-(b) where population sizes correspond roughly to 33%, 66%, and 133% of the POS for 3 objectives and 33% and 66% for 5 and 6 objectives, as shown in Table 1. On the contrary, IBEA finds more solutions than NSGA-II when population sizes are relatively a small fraction of the POS. See Fig.1 (c)-(d) where population sizes $\{50, 100, 200\}$ are used in 5 and 6 objectives, which correspond to fractions in the range 0.3%–3.2% of the POS. In 4 objectives, Fig.1 (b), an interesting transition

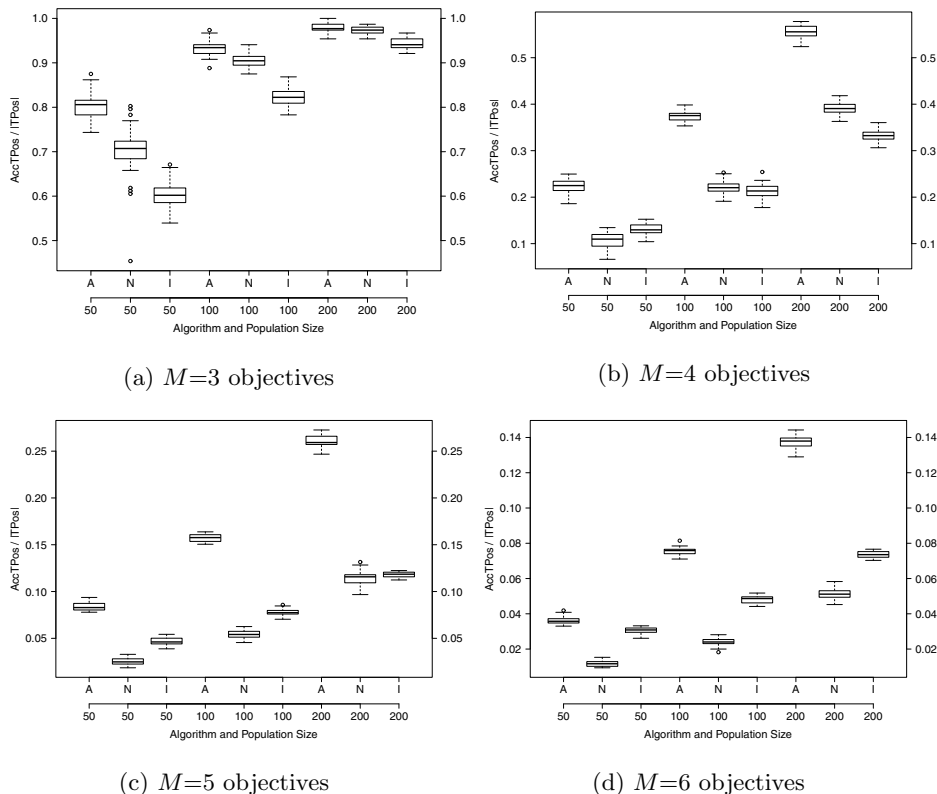
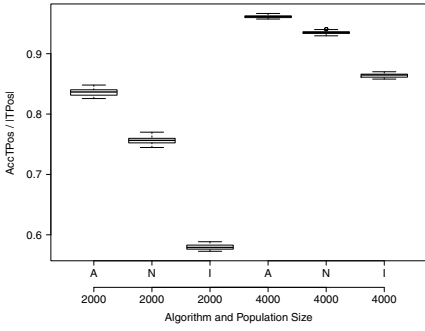


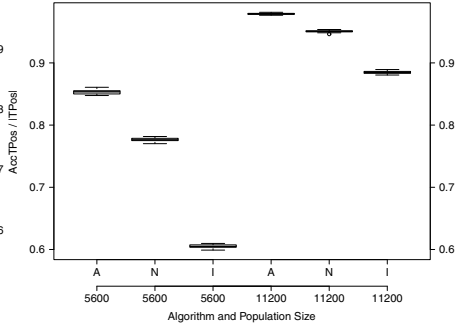
Fig. 1. Resolution of the approximation at the end of the run $\alpha(T)$, i.e. ratio of accumulated number of Pareto optimal solutions found to the size of the POS. Population sizes 50, 100, and 200 for 3, 4, 5, and 6 objectives. Algorithms $A\epsilon S\epsilon H$ (A), NSGA-II (N) and IBEA (I).

can be observed. When the smallest population is used, i.e. 50 individuals $\sim 3.2\%$ of POS, IBEA finds more solutions than NSGA-II. For a population size of 100 $\sim 6.4\%$ of POS NSGA-II finds a slightly larger number of solutions than IBEA. For a population size of 200 $\sim 12.9\%$ of POS, NSGA-II finds a significant larger number of solutions than IBEA.

The gap between $A\epsilon S\epsilon H$ and the other two algorithms augments when the population size increases within a range in which it still is a small fraction of the POS, as shown in Fig.1 (b)-(d) where the ranges in which population size increase are 3.2% – 12.9%, 0.8% – 3.2%, and 0.3% – 1.2% of POS for 4, 5, and 6 objectives, respectively. On the other hand, the gap reduces when population size increases within a range in which it is a large fraction of the POS, as shown in Fig.1 (a) and Fig.2 (a)-(b) where the ranges in which population size increase are roughly 33% – 133% of POS for 3 objectives and 33% – 66% for 5 and 6 objectives.

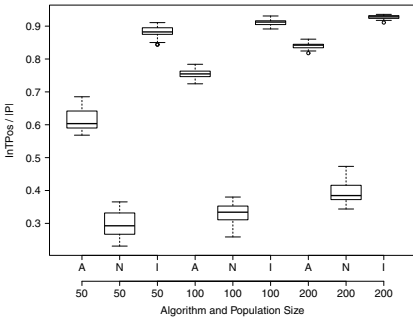


(a) $M=5$ objectives

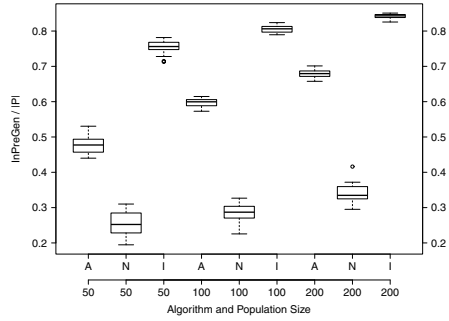


(b) $M=6$ objectives

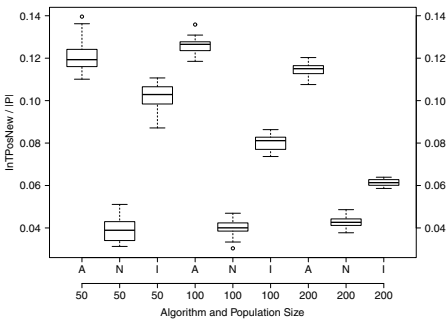
Fig. 2. Ratio of accumulated number of Pareto optimal solutions found to the size of the POS. Population sizes $\{500, 1000\}$, $\{2000, 4000\}$, and $\{5600, 11200\}$ for 4, 5, and 6 objectives, respectively. Algorithms $A\epsilon S\epsilon H$ (A), NSGA-II (N) and IBEA (I).



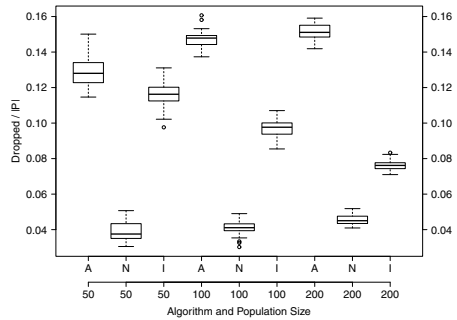
(a) $\bar{\tau}$: Pareto optimal



(b) $\bar{\tau}^-$: Old Pareto optimal



(c) $\bar{\tau}^*$: Absolutely new Pareto optimal



(d) $\bar{\delta}$: Dropped Pareto optimal

Fig. 3. Boxplots of average generational search-assessment indices in 30 runs. Population sizes $\{50, 100, 200\}$, 6 objectives, $T = 100$ generations. Algorithms $A\epsilon S\epsilon H$ (A), NSGA-II (N) and IBEA (I).

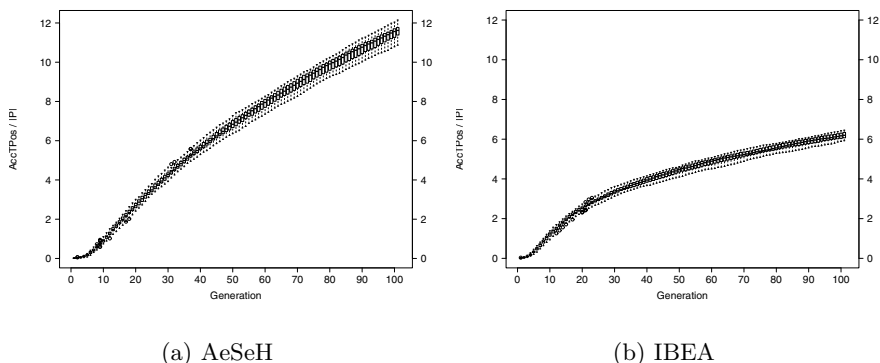


Fig. 4. Accumulated population gains $\beta(t)$ over the generations. Population size 200, 6 objectives. Algorithms AeSeH and IBEA.

4.3 Generational Search Assessment Indices

Fig.3 (a)-(d) show boxplots of some \bar{I} indexes computed from data obtained in 30 independent runs of the algorithms iterating $T = 100$ generations. Results are shown for 6 objectives landscapes using population sizes $\{50, 100, 200\}$, which are relatively small compared to the POS. From these figures important observations are as follow.

In average, at each generation, IBEA contains in its population a very large number of PO solutions compared to AeSeH and NSGA-II as shown in Fig.3 (a). Note that the median of index $\bar{\tau}$ for IBEA is in the range 0.88-0.95, whereas the ranges for AeSeH and NSGA-II are between 0.6-0.85 and 0.3-0.4, respectively.

However, the number of old PO solutions (PO solutions present in the current population and also in the population of the previous generation) for IBEA is much larger than for AeSeH and NSGA-II, as shown in Fig.3 (b). Note that the median of $\bar{\tau}^-$ is in the range 0.75-0.85 for IBEA, whereas $\bar{\tau}^-$ is in the range 0.48-0.68 for AeSeH and 0.25-0.35 for NSGA-II.

In fact, the generational average number of absolutely new PO solutions (PO solutions in the current population that have not been discovered in previous generations) is larger for AeSeH than for IBEA and NSGA-II, as shown in Fig.3 (c). Note that the median of index $\bar{\tau}^*$ for AeSeH is around 0.12, whereas for IBEA it reduces with population size from 0.10 to 0.06 and slightly increases for NSGA-II from 0.039 to 0.04. The similar $\bar{\tau}^*$ values by AeSeH are a good sign of robustness to population size variations, i.e. a similar discovery rate could be expected with various population sizes. On the contrary, IBEA's discovery rate could reduce significantly with population size. If the evaluation of the algorithms is done based only on the points included in the population at a given generation, as it is often the case, IBEA is likely to contain more PO solutions than AeSeH, as shown in Fig.3 (a), and therefore be considered a better algorithm. However,

A ϵ S ϵ H finds twice as many PO solutions than IBEA, as shown in Fig.1 (d). These results show that IBEA could be a good algorithm for finding a low resolution approximation of the POS, but for high resolutions is not efficient. In general, these results show the importance of properly evaluating the algorithms according to the aim of the optimization task at hand.

The index of dropped PO solutions $\bar{\delta}$ (PO solutions present in the population of the previous generation that are not included in the current population after truncation selection) shows a trend very similar to the one observed for the index $\bar{\tau}^*$, as shown in Fig.3 (d). Dropping superior solutions in favor of solutions that appear non-dominated in the population but are inferior in the landscape could be seen as a selection weakness. However, this could also be a source of exploration. This deserves further research.

The accumulated population gains $\beta(t)$ for A ϵ S ϵ H and IBEA are illustrated in Fig.4 for population size 200 and 6 objectives. Note that just after 20 generation the gain by A ϵ S ϵ H is already larger than by IBEA. At the end of the run, A ϵ S ϵ H is able to generate an approximation twelve times the size of its population, whereas IBEA is able to generate an approximation 6 times the size of its population.

5 Conclusions

This work has studied the behavior of NSGA-II, IBEA and A ϵ S ϵ H generating a high-resolution approximation of the POS. The study has clarified the ability and efficiency of the algorithms assuming scenarios where it is relatively easy to hit the POS, showing the importance to properly assess algorithm's performance according to the task of the optimizer in many objective optimization. In the future, we would like to extend our study to larger landscapes in order to understand the behavior of selection in scenarios where the convergence ability towards the POS is determinant to achieve a good resolution. Also, we would like to study other indicators for IBEA and other many-objective algorithms.

References

1. Aguirre, H., Tanaka, K.: Insights on Properties of Multi-objective MNK-Landscapes. In: Proc. 2004 IEEE Congress on Evolutionary Computation, pp. 196–203. IEEE Service Center (2004)
2. Nishio, Y., Oyama, A., Akimoto, Y., Aguirre, H., Tanaka, K.: Many-Objective Optimization of Trajectory Design for DESTINY Mission. In: Learning and Intelligent Optimization Conference. LNCS. Springer (2014)
3. Hadka, D., Reed, P.: Borg: An Auto-adaptive Many-objective Evolutionary Computing Framework. *Evol. Computation* 2(2), 231–259 (2013)
4. Aguirre, H., Oyama, A., Tanaka, K.: Adaptive ϵ -Sampling and ϵ -Hood for Evolutionary Many-Objective Optimization. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 322–336. Springer, Heidelberg (2013)

5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II., KanGAL report 200001 (2000)
6. Zitzler, E., Künzli, S.: Indicator-based Selection in Multiobjective Search. In: Yao, X., et al. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)

A Multiobjective Evolutionary Optimization Framework for Protein Purification Process Design

Richard Allmendinger and Suzanne S. Farid

Department of Biochemical Engineering, University College London,
Torrington Place, London WC1E 7JE, UK
{r.allmendinger, s.farid}@ucl.ac.uk

Abstract. Increasing demand in therapeutic drugs has resulted in the need to design cost-effective, flexible and robust manufacturing processes capable of meeting regulatory product purity requirements. To facilitate this design procedure, a framework linking an evolutionary multiobjective algorithm (EMOA) with a biomanufacturing process economics model is presented. The EMOA is tuned to discover sequences of chromatographic purification steps, and equipment sizing strategies adopted at each step, that provide the best trade-off with respect to multiple objectives including cost of goods per gram (COG/g), robustness in COG/g, and impurity removal capabilities. The framework also simulates and optimizes subject to various process uncertainties and design constraints. Experiments on an industrially-relevant case study showed that the EMOA is able to discover purification processes that outperform the industrial standard, and revealed several interesting trade-offs between the objectives.

1 Introduction

The biotech sector is facing increasing pressures to design more cost-efficient, robust and flexible manufacturing processes [1]. Among biotech therapies, monoclonal antibodies (mAbs) represent one of the fastest growing category due to their unique binding specificity to targets. A typical antibody purification process is depicted in Figure 1: in upstream processing (USP) mammalian cells expressing the mAb of interest are cultured in bioreactors, whilst in downstream processing (DSP) the mAb is recovered, purified and cleared from viruses using a variety of operations. Of these steps, chromatography operations are identified as critical steps and can represent a significant proportion of the purification material costs. The design of cost-effective purification processes can help addressing this challenge.

The design stage is further complicated by the fact that regulatory bodies expect biopharmaceutical companies to fully understand their manufacturing process, thus account for uncertainty in the manufacturing process, and be able to establish a purification process that conforms to strict purity requirements. To assist the process of tackling these challenges, presented here is an optimization-based framework linking an evolutionary multiobjective optimization algorithm (EMOA) with a biomanufacturing process economics model. The goal of the EMOA is to discover sequences of chromatographic purification steps, and sizing strategies adopted at each step, that provide the best trade-off with respect to multiple objectives including cost of goods per

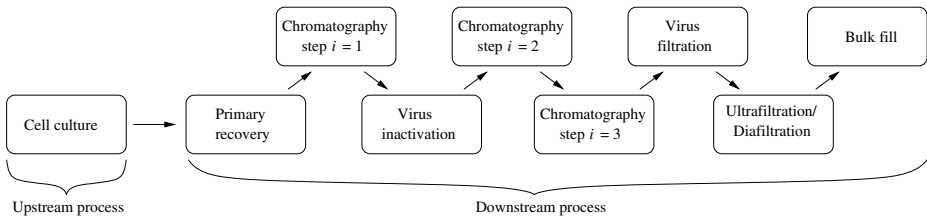


Fig. 1. Typical flowsheet for an antibody manufacturing process

gram (COG/g), robustness in COG/g, and impurity removal capabilities. The objectives are then computed by the process economics model serving as the fitness evaluation tool. Additional complexities accounted for by the framework include simulating and optimizing subject to uncertainty and constraints.

This paper extends our previous work on chromatography design/optimization [2,3], which assumed a fixed sequence of chromatography steps, and focused on tuning (using a single-objective EA) the chromatography column sizing adopted at each step such that the COG/g are minimized only. This extension posed two challenges including (i) the development of a customized EMOA accounting for constraints and variables of different type, and (ii) the extension of the process economics model so as to account for additional design choices and their impact on manufacturing performance.

Chromatography design/optimization can be tackled from several other angles. For example, the exploration of non-Protein A based purification processes was considered by Chen et al. [4]. Tuning of chromatographic operating conditions is another prominent research field [5], and so is resin screening [6]. Unlike our simulation-based work, these studies are based on real physical experiments. A simulation-based approach was also adopted by Liu et al. [7], where mathematical programming is proposed to address chromatography column sizing and sequencing in the context of biopharmaceutical facility design. Stonier et al. [8] proposed a discrete-event simulation for the selection of optimal chromatography column diameters over a range of titres.

The application of multiobjective optimization to chromatography design/ optimization has become popular only recently. For example, Nfor et al. [9] used EMO to tune operating parameters (e.g. column loading, flowrate and gradient length) of a single chromatography step so as to improve recovery yield, purity, and productivity. The focus in this paper is on optimizing “high-level” criteria relating to all chromatography steps (e.g. impurity removal capabilities) or the complete manufacturing process (e.g. COG/g and its robustness). Moreover, uncertainty is associated with global operating parameters (e.g. product titre and initial impurity levels) as well as with chromatography specific parameters (e.g. step yields and step specific removal capabilities).

2 Constrained Multiobjective Purification Process Design

The framework proposed is based on the following closed-loop: an EMOA creates solutions \mathbf{x} (i.e. a sequence of chromatography steps and column sizing strategies), which are then decoded, embedded into a feasible manufacturing process (see Figure 1),

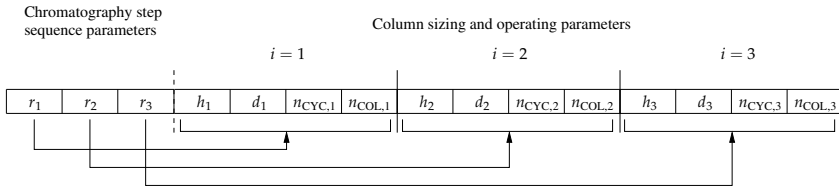


Fig. 2. Representation of a candidate solution for $k = 3$ chromatography steps. Each step $i = 1, \dots, k$ is defined by a resin r_i and a column sizing strategy, which is composed of the bed height h_i and diameter d_i of columns, number of cycles $n_{CYC,i}$ each column is used, and the number of columns $n_{COL,i}$ operating in parallel.

and evaluated by a biomanufacturing process economics model; manufacturing uncertainties are accounted for using Monte Carlo (MC) trials. Objective values pertaining to \mathbf{x} are recorded and fed back to the EMOA to be considered in the generation of future solutions. The decision variables, constraints, objective functions, and uncertain factors the problem is subject to are explained in more detail in the following.

Decision Variables: Figure 2 shows the string encoding developed to represent a purification process or solution \mathbf{x} . Assuming a fixed number of chromatographic steps k , the task is to define, for each step $i = 1, \dots, k$, the resin $r_i \in \{\text{resin}_1, \dots, \text{resin}_q\}$ and column sizing strategy, which is composed of the bed height h_i and diameter d_i of a column, number of cycles each column is used for $n_{CYC,i}$, and the number of columns operating in parallel $n_{COL,i}$. Therefore, the problem is subject to $l = k + 4 \cdot k$ variables in total. The choice of the resin r_i used dictates several chromatographic operation and cost parameters considered by the biomanufacturing process economics model, such as the step yield, resin price, and impurity removal capabilities. On the other hand, the sizing strategy adopted at each chromatographic step i defines the total volume of resin V_i available, and the processing time T_i that the chromatography step take; T_i and V_i are calculated as follows [10]:

$$V_i = \pi \cdot d_i^2 / 4 \cdot h_i \cdot n_{CYC,i} \cdot n_{COL,i} \tag{1}$$

$$T_i = n_{CYC,i} \cdot h_i \cdot (CV_{BUFF,i} + CV_{LOAD,i} / n_{COL,i}) \cdot u_i, \tag{2}$$

where $CV_{BUFF,i}$ and $CV_{LOAD,i}$ are the number of column volumes of buffer and product load per cycle, and u_i is the linear velocity of resin r_i .

Constraints: The problem is subject to three types of constraints:

1. *Chromatography sequence constraints* are defined on the variables $r_i, i = 1, \dots, k$ and ensure that a purification process consists of non-identical, feasible and orthogonal (i.e. different typed) chromatography steps, or more formally

$$g_1 : r_i \neq r_j, \quad i, j = 1, \dots, k, \quad i \neq j, \tag{3}$$

$$g_2 : r_i^i = 1, \quad i = 1, \dots, k, \tag{4}$$

$$g_3 : r_i^T \neq r_j^T, \quad i, j = 1, \dots, k, \tag{5}$$

where r_i^T denotes the resin type of r_i , and r_i^i is a boolean variable indicating whether resin r_i is permitted to be used at position i .

2. The *demand constraint* ensures that the annual amount of product manufactured P is sufficient to satisfy the annual demand D , or $g_4 : P \geq D$.
3. *Resin requirement constraints* act on the column sizing variables and ensure that, at each step $i = 1, \dots, k$, there is sufficient resin volume V_i available to process the mass of product M_i coming in from the previous unit operation. Formally, this constraint can be defined as

$$g_5 : V_i \geq \frac{M_i}{r_{i,DBC} \cdot \kappa} \quad i = 1, \dots, k, \quad (6)$$

where V_i is computed according to Equation (2), $r_{i,DBC}$ is the DBC of the resin used at step i , and $0 < \kappa \leq 1$ the maximum capacity utilization factor.

Manufacturing Uncertainties: Several uncertain factors arising in the manufacturing process are captured by the framework: (i) product titre, (ii) chromatography step yields, (iii) DBC, (iv) eluate volumes, (v) HCP log reduction, and (vi) initial HCP level. While uncertainties in (i) and (vi) are due to fluctuations arising in USP, the other factors are associated with the resins r_i available for selection and sensitivity of operating conditions. Uncertainties are modeled by associating each factor with a probability distribution (reflecting real-world variability) from which values are drawn at random during Monte Carlo (MC) trials; the way the data resulting from the MC trials is processed by the EMOA will be detailed in the next section.

Performance Metrics: Three objectives are considered to drive the search for cost-efficient and reliable purification process yielding highly pure products:

1. The *cost of goods per gram* $COG/g = C/P$, where C is the sum of annual direct costs (e.g. consumables and labor) and indirect costs (e.g. capital charge and facility-related costs) and P the annual product output, represent the costs for manufacturing a single gram of product and are to be *minimized*.
2. The *robustness in COG/g*, η , is defined here as the ratio

$$\eta = \frac{\sigma_{COG/g}}{\mu_{COG/g}} = \frac{\sqrt{\frac{1}{N} \sum_{j=1}^N (COG/g_j - \mu_{COG/g})^2}}{\mu_{COG/g}}, \quad (7)$$

where N is the number of MC trials performed for a specific process so far, COG/g_j the COG/g value at MC trial j , and $\mu_{COG/g} = \frac{1}{N} \sum_{j=1}^N COG/g_j$. The smaller the value of η , the less variation there is in COG/g in the presence of uncertainty. Hence, the objective is to *minimize* η .

3. The *probability of meeting purity requirements* $p(\text{meeting required purity})$ is the probability that a purification process reduces the HCP impurity level in a product below a certain limit HCP^* . This probability is to be *maximized* and computed here by

$$p(\text{meeting required purity}) = \frac{1}{N} \sum_{i=1}^N \delta_i, \quad \text{where } \delta_i = \begin{cases} 1 & \text{if } HCP_{Final_i} < HCP^* \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

where N is the number of MC trials performed, and HCP_{Final_i} the final HCP level at MC trial i . The final HCP level is calculated here by $HCP_{Final_i} = HCP_{initial} / 10^{\sum_{i=1}^k r_{i,HCP}}$, where $r_{i,HCP}$ is the HCP log reduction of resin r_i .

All objectives are obtained by running the process economics model, which is based on mass balance and cost equations as defined in [10]. Note, whilst COG/g is a standard metric, the metrics, p (meeting required purity) and η , have not been considered in the literature so far.

3 Experimental Setup

This section describes the case study, EMOA and its parameter settings as used in the subsequent experimental analysis.

Case Study Setup: The case study was adopted from [3] and focuses on a single-product mAb manufacturing facility that employs a process flowsheet as shown in Figure 1 with $k = 3$ chromatography steps. Assumed is an annual demand of $D = 400\text{kg}$, a product titre of 3g/L , and a desired final HCP level of $HCP^* = 100\text{ng/mg}$ (which is typical of final product specification limits for recombinant proteins). Two initial HCP levels are investigated, $HCP_{initial} = \{10^5, 10^6\}\text{ng/mg}$.

A total of $q = 10$ resins, comprising around 125 different sequences, are available for tuning the sequence of chromatography steps. For the characteristics of these resins, and technical details of the manufacturing process and resource cost assumptions please refer to [2]. Table 1 lists the uncertain parameters and their common levels of uncertainty in the context of triangular probability distributions; i.e. a variation of $x\%$ corresponds to the distribution $\text{Tr}(x \cdot (100 - x)/100, x, x \cdot (100 + x)/100)$. The value range of column sizing parameters is $15\text{cm} \leq h_i \leq 25\text{cm}$ (11 values), $50\text{cm} \leq d_i \leq 200\text{cm}$ (10 values), $n_{\text{CYC},i} \in \{1, \dots, 10\}\text{cm}$, $n_{\text{COL},i} \in \{1, \dots, 4\}$, $i = 1, 2, 3$; i.e. in total the search space comprises $(11 \cdot 10 \cdot 10 \cdot 4)^3 \cdot 125 \approx 10.5 \cdot 10^{12}$ different purification processes. The industrial platform employs a fixed and commonly used chromatography step sequence, $PrA \rightarrow L \rightarrow CEX \rightarrow L \rightarrow AEX$, in combination with the sizing strategy (which is set based on empirical rules) $n_{\text{COL},i} = 1, h_i = 20\text{cm}, n_{\text{CYC},i} = 5, i = 1, 2, 3$, with d_i being adjusted such that the resulting total resin volume V_i (Equation (2)) satisfies the resin requirement constraint (Equation (6)).

Evolutionary Multiobjective Optimization: The focus in this work is to understand how EMO can be tuned to tackle the purification process design problem rather than comparing different EMOAs. Hence, to guide the search, the popular NSGA-II [11] is extended with methods for coping with the model uncertainties and constraints, which are explained below. The algorithm uses binary tournament selection, uniform crossover, and a mutation operator that selects a random value from the range of possible values.

Constraint-Handling Strategies: The *chromatography sequence constraints* (Equations (3) to (5)) are addressed by programming the sequence-related variables $r_i, i = 1, \dots, k$ as a single variable S representing all feasible sequences. For population initialization, a sequence is selected at random from S . Crossover and mutation are applied directly on the variables r_i but resulting infeasible offspring are repaired by selecting a sequence from S that differs in as few steps $i = 1, \dots, k$ as possible from the

Table 1. Probability distributions associated with uncertain factors

<i>Uncertain factor</i>	<i>Variation (%)</i>
Product titre	13.3
Chrom. step yields	5
DBC	10
Eluate volume	10
HCP logs	20
Initial HCP level	20

Table 2. Default parameter settings of the EMOA

<i>Parameter</i>	<i>Setting</i>
Population size μ	50
Per-variable mutation probability	1/1
Crossover probability	0.6
Number of generations G	50
Monte Carlo trials N	100

original sequence. Ties between equally close sequences are broken at random. The *demand constraint* is circumvented by setting up USP such that there is a slight product surplus. To cope with the *resin requirement constraint* (Equation (6)), a ‘repairing’ strategy is employed that iteratively increases the values of the column-sizing related variables (associated with a particular chromatography step i), one variable at the time, until sufficient resin is available (i.e. until Equation (6) is satisfied) or until the maximum value of a variable is reached, in which case the value of another variable is increased. The sequence in which variables are modified affects the performance of the optimizer as indicated in [3] for the single-objective case. The default sequence adopted is $d_i \rightarrow n_{CYC,i} \rightarrow h_i \rightarrow n_{COL,i}$, which performed best in [3], but alternative sequences will be considered in the experimental study.

Uncertainty-Handling Strategy: Model uncertainties are accounted for by exposing a manufacturing process to N MC trials with values of uncertain factors being drawn at each trial from the probability distribution associated with the factors. The objective values of a solution were then the averages of the different performance metrics across the N trials, and these averages were updated if the same solution is evaluated multiple times during the search.

The experimental study presents a sensitivity analysis of the performance metrics, and investigates the robustness of the EMOA using the proposed framework. The default settings of the EMOA are given in Table 2. To allow for fair comparison of processes discovered by the EMOA, all processes present in the final population are evaluated using 1000 MC trials. Any results shown are average results across 30 independent algorithm runs.

4 Experimental Study

Sensitivity Analysis to Identify Global Drivers of Cost and Purity: Figure 3 uses the idea of tornado plots to show the impact of several uncertain factors on COG/g and the final HCP level HCP_{Final} . The boxplots have been created based on 10000 randomly generated, feasible and unique purification processes. For each process, plotted is the overall maximal effect on the two metrics of the best and worst case setting of the uncertain factors. It can be observed from the plots that generally the impact of model parameters depends on the objective being optimized, and increasing the number of parameters affects performance more significantly. There seems to be a symmetric

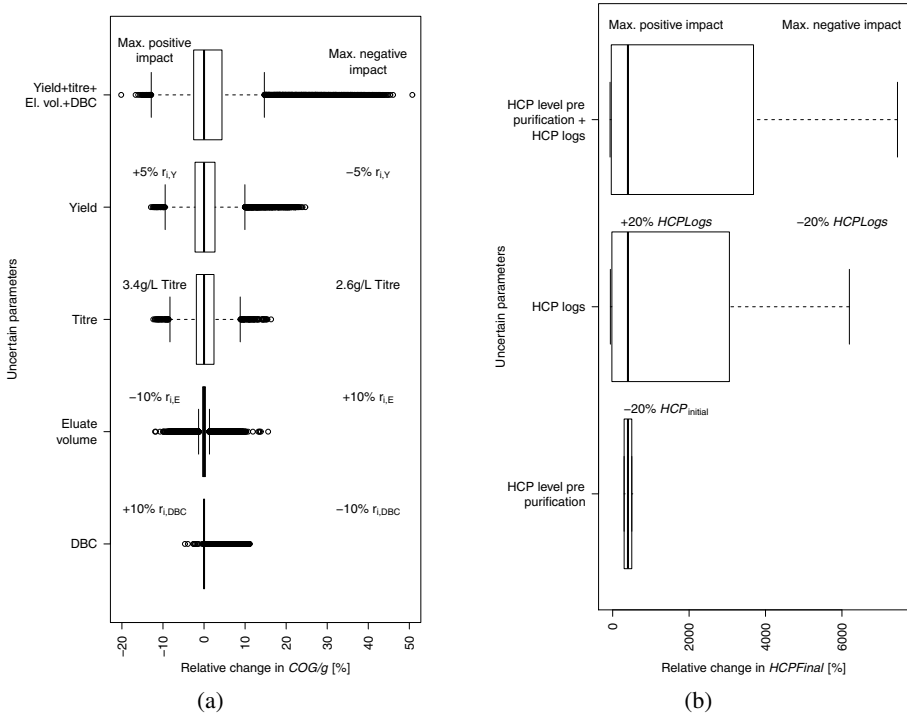


Fig. 3. Tornado diagrams illustrating the overall maximal effects of best (left of zero) and worst (right of zero) case settings of uncertain factors — DBC, elution volume, titre, and yield (bottom four boxplots in (a)) and HCP log reductions and initial HCP level (bottom two boxplots in (b)), and all four, respectively, two parameters at once (top boxplot) — on (a) COG/g and (b) final HCP levels HCP_{Final}

positive and negative impact of the uncertain factors on the objective COG/g (Figure 3(a)). The step yield and titre are most sensitive to uncertainty as they have a direct impact on the mass of product manufactured (and thus the denominator of the metric COG/g). Uncertainties in initial HCP levels and HCP log reductions are the only factors that affect the final HCP level HCP_{Final} (Figure 3(b)). The negative effect on HCP_{Final} is significantly greater than the positive effect because many of the (randomly generated) purification processes are able to reduce the HCP level down to $HCP_{Final} \approx 0$ (though this might be associated with high COG/g), leaving limited scope for further improvements. Note, a reduction in HCP_{Final} translates into an increase in p (meeting required purity).

Tuning an EMOA to Cope with Uncertainty and Constraints: This section gives a taste of how the discovery process of optimal purification processes can be affected by the choice of algorithm parameter settings. Figure 4 uses the concept of (median) attainment surfaces [12] to visualize the typical convergence behavior of the EMOA (top left plot) and the performance impact on the EMOA by two algorithm settings,

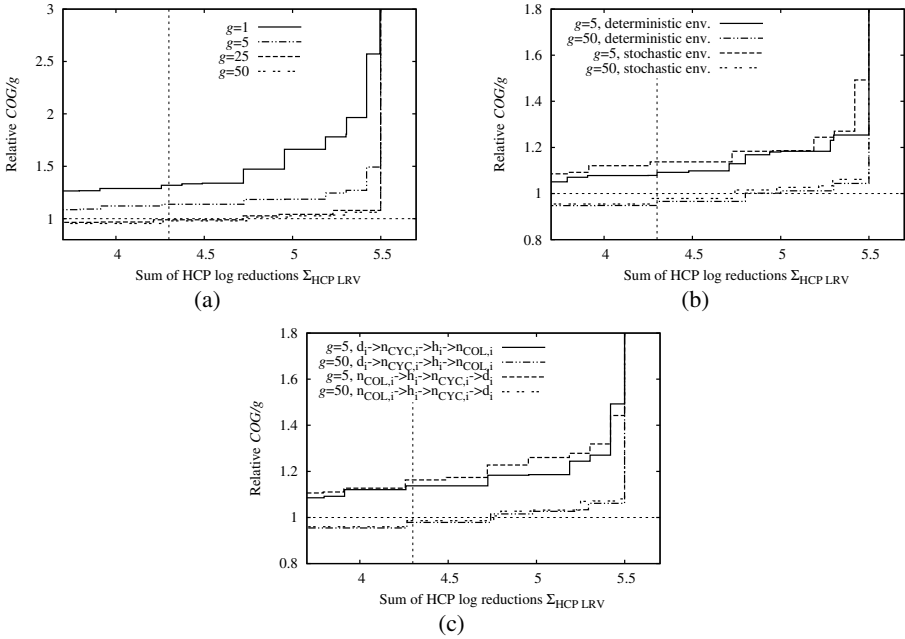


Fig. 4. Median attainment surfaces (MASs) obtained by an EMOA optimizing COG/g and the sum of HCP log reductions $\Sigma_{HCP,LRV} = \sum_{i=1}^k r_{i,HCP}$. The plots show (a) the MASs at different generations g with uncertainty, (b) MASs for different g obtained in a deterministic and stochastic environment, and (c) the MASs for different g and repairing strategies with uncertainty. The performance of the industrial platform is indicated by the dashed horizontal and vertical lines.

the number of MC trials N (top right plot) and the constraint-handling strategy (bottom plot). In all three plots, the EMOA minimized the COG/g and the sum of HCP log reductions $\sum_{i=1}^k r_{i,HCP}$.

From Figure 4(a) it can be seen that the EMOA needs to be run for around $g \approx 25$ generations to match and outperform the industrial platform. Comparing the convergence speed and final solution quality obtained by the EMOA with and without uncertainty (Figure 4(b)), it is apparent that uncertainty harms both aspects significantly. Figure 4(c) shows that the constraint-handling strategy adopted is crucial too. In fact, repairing according to the scheme $d_i \rightarrow n_{CYC,i} \rightarrow h_i \rightarrow n_{COL,i}$ yields best results as increasing the column diameter d_i first is often sufficient to satisfy the resin requirement constraint without sacrificing processing time significantly.

EMO Applied to all Three Objectives Subject to Uncertainty: Figure 5 uses heatmaps to visualize the trade-off between all three objectives for two HCP levels $HCP_{initial} = 10^5 \text{ng/mg}$ (Figure 5(a)) and 10^6ng/mg (Figure 5(b)). Several trade-offs can be observed from the plots: (i) the range of the metric p (meeting required purity) increases with the initial HCP level, (ii) the COG/g increases as p (meeting required purity) increases and/or η decreases, and (iii) an improvement in the robustness η is achieved by adopting smaller column dimensions (supporting figure not shown here). The heatmaps can also be exploited to make design decisions. For example, assume that the goal of a

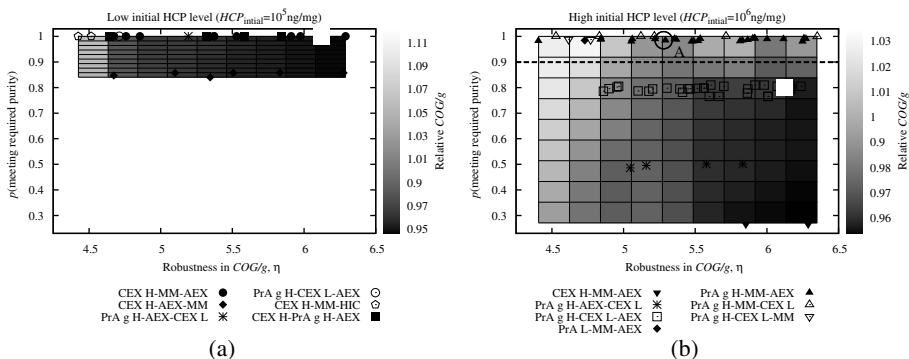


Fig. 5. (Pareto) optimal purification processes discovered by an EMOA optimizing three objectives, COG/g , $p(\text{meeting required purity})$, and η , for two different HCP levels $HCP_{\text{initial}} = 10^5 \text{ ng/mg}$ (a) and 10^6 ng/mg (b). The performance of the industrial platform is indicated by the big white square in each plot. The response surface was generated by interpolating the processes' objective values using the Kriging function, $\text{Krig}()$, from the *fields* package of the statistical software R.

manufacturer is to establish a process with $p(\text{meeting required purity}) > 0.9\%$. Whilst in this case there is no incentive to deviate from the industrial platform from the perspective of COG/g and $p(\text{meeting required purity})$ for a low initial HCP level, a different sequence is needed for a high initial HCP level. For instance, the sequence $PrAg H \rightarrow MM \rightarrow AEX$, as indicated by the letter A in Figure 5(b), meets the purity requirements without increasing the COG/g significantly.

5 Summary and Conclusion

Presented was a framework for designing cost-efficient and robust chromatographic purification process that yield pure biopharmaceuticals. The framework comprised a process economics model and an EMOA, which optimized the sequence of chromatography steps and column sizing strategies with respect to multiple objectives and subject to uncertainty. Validating the framework on an industrially-relevant case study revealed that the performance impact of an uncertain factor depends on the objective being optimized. This knowledge can be used e.g. to diagnose which process parameters need a tighter control. Furthermore, the framework was able to discover purification processes that outperform the industrial platform, and revealed interesting trade-offs between objectives that can facilitate the design of purification process. Future research will look at extending the framework to cover additional design choices and investigate more efficient uncertainty-handling strategies.

References

1. Kelley, B.: Industrialization of mab production technology: The bioprocessing industry at a crossroads. *mAbs* 1, 443–452 (2009)
2. Simaria, A.S., Turner, R., Farid, S.S.: A multi-level meta-heuristic algorithm for the optimisation of antibody purification processes. *Biochemical Engineering Journal* 69, 144–154 (2012)

3. Allmendinger, R., Simaria, A.S., Turner, R., Farid, S.S.: Closed-loop optimization of chromatography column sizing strategies in biopharmaceutical manufacture. *Journal of Chemical Technology and Biotechnology* (2013), doi:10.1002/jctb.4267
4. Chen, J., Tetrault, J., Zhang, Y., Wasserman, A., Conley, G., DiLeo, M., Haimes, E., Nixon, A.E., Ley, A.: The distinctive separation attributes of mixed-mode resins and their application in monoclonal antibody downstream purification process. *Journal of Chromatography A* 1217(2), 216–224 (2010)
5. Treier, K., Berg, A., Diederich, P., Lang, K., Osberghaus, A., Dismar, F., Hubbuch, J.: Examination of a genetic algorithm for the application in high-throughput downstream process development. *Biotechnology Journal* 7, 1203–1215 (2012)
6. Rathore, A.S.: Resin screening to optimize chromatographic separations. *LC–GC* 19(6), 616–622 (2001)
7. Liu, S., Simaria, A.S., Farid, S.S., Papageorgiou, L.G.: Designing cost-effective biopharmaceutical facilities using mixed-integer optimization. *Biotechnology Progress* 29(6), 1472–1483 (2013)
8. Stonier, A., Smith, M., Hutchinson, N., Farid, S.S.: Dynamic simulation framework for design of lean biopharmaceutical manufacturing operations. *Computer Aided Chemical Engineering* 26, 1069–1073 (2009)
9. Nfor, B.K., Zuluaga, D.S., Verheijen, P.J.T., Verhaert, P.D.E.M., van der Wielen, L.A.M., Otens, M.: Model-based rational strategy for chromatographic resin selection. *Biotechnology Progress* 27(6), 1629–1643 (2001)
10. Farid, S.S., Washbrook, J., Titchener-Hooker, N.J.: Modelling biopharmaceutical manufacture: Design and implementation of SimBiopharma. *Computers & Chemical Engineering* 31, 1141–1158 (2007)
11. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
12. Fonseca, C.M., Fleming, P.J.: On the performance assessment and comparison of stochastic multiobjective optimizers. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) *PPSN IV. LNCS*, vol. 1141, pp. 584–593. Springer, Heidelberg (1996)

Automatic Design of Evolutionary Algorithms for Multi-Objective Combinatorial Optimization

Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle

IRIDIA, Université Libre de Bruxelles (ULB), Brussels, Belgium
{lleonaci,manuel.lopez-ibanez,stuetzle}@ulb.ac.be

Abstract. Multi-objective evolutionary algorithms (MOEAs) have been the subject of a large research effort over the past two decades. Traditionally, these MOEAs have been seen as monolithic units, and their study was focused on comparing them as blackboxes. More recently, a component-wise view of MOEAs has emerged, with flexible frameworks combining algorithmic components from different MOEAs. The number of available algorithmic components is large, though, and an algorithm designer working on a specific application cannot analyze all possible combinations. In this paper, we investigate the automatic design of MOEAs, extending previous work on other multi-objective metaheuristics. We conduct our tests on four variants of the permutation flowshop problem that differ on the number and nature of the objectives they consider. Moreover, given the different characteristics of the variants, we also investigate the performance of an automatic MOEA designed for the multi-objective PFSP in general. Our results show that the automatically designed MOEAs are able to outperform six traditional MOEAs, confirming the importance and efficiency of this design methodology.

1 Introduction

Multi-objective evolutionary algorithms (MOEAs) are the most studied metaheuristics for solving multi-objective optimization problems and a large number of MOEAs have been proposed in the past two decades [1, 2, 8, 10, 21, 22]. A few surveys of the field have been conducted [19, 24] and competitions have been held to identify the best MOEA for particular benchmarks [20]. These works study MOEAs as monolithic units and provide insights on which particular MOEAs are state of the art for specific problems, giving a baseline for future developments. More recently, a component-wise view of MOEAs has drawn the attention of the MOEA community [5, 13]. “Deconstructed” MOEAs actually differ by a few main algorithmic components, which can be individually analyzed to assess their actual contribution to the overall efficiency of the algorithm. This component-wise view has been recently strengthened by the development of flexible algorithmic frameworks [11, 13], where novel MOEAs can be devised combining existing algorithmic components. However, the potential of such approach remains unclear as the number of possible combinations is extremely large to be fully explored.

In a parallel research trend, the automatic design of multi-objective optimizers has produced several state-of-the-art algorithms [3, 9, 16]. This methodology

consists of applying automatic offline parameter configuration tools (*tuners*, for short) to flexible algorithmic frameworks, such as the ones recently proposed for MOEAs. By doing so, the parameter space searched by the tuner is actually a space of different algorithms, which allows the analysis and comparison of several novel algorithms at a time. The final configuration selected by the tuner is not necessarily the best possible algorithm as the configuration space is not searched exhaustively, but it is usually a high-performing one. In this work, we investigate the efficiency of the automatic algorithm design methodology for creating MOEAs for combinatorial optimization problems. We use four different variants of the permutation flowshop problem (PFSP) as test benchmarks. These variants combine, in different ways, the most relevant optimization criteria for the PFSP, namely *makespan* (C_{\max}), *total flow time* (TFT), and *total tardiness* (TT).

In a first step, we present the framework we use for this work, which combines components from ParadisEO-MOEO [13], PISA [7], and PaGMO [6]. Particularly, some of the MOEAs from the literature cannot be easily represented just by the combination of a fitness and a diversity component as proposed in ParadisEO-MOEO. Instead, following [23], we use a more general preference relation, defined as a combination of a set-partitioning criterion, a quality indicator and a diversity measure. Second, although some MOEAs claim to use an external archive, this archive is used during the evolutionary search process, either for mating or selection. In our implementation, we allow MOEAs to use two archives at a time: an internal one, which can replace the population of the algorithm; and an external one, which is only used for building the final approximation front. Finally, our implementation incorporates elements from new MOEAs, such as sequential versus one-shot removal of solutions [1], quality in terms of hypervolume contribution [1, 2], and adaptive-grid diversity [12].

In a second step, we generate five automatically designed MOEAs (AutoMOEAs): one for each of the PFSP variants considered here, and an extra one for solving all variants (AutoMOEA_{PFSP}). Since related works have shown these variants present different search space characteristics [5, 9], it is important to understand whether (i) the automatic design methodology can produce a single optimizer that performs better than the others for all variants, or; (ii) the best strategy is to have an AutoMOEA tailored for each variant. We then conduct an experimental analysis on a PFSP variant basis. For each PFSP variant, we (i) compare the structure of the variant-specific AutoMOEA to AutoMOEA_{PFSP}; and (ii) compare their performance to six traditional MOEAs. Results show that the AutoMOEAs are often able to outperform the traditional MOEAs. Although the more general AutoMOEA_{PFSP} reaches better results than the traditional MOEAs, the variant-specific AutoMOEAs perform better in most scenarios, reinforcing the need for the automatic algorithm design methodology.

2 A Framework for Instantiating MOEAs

A summary of the components that differentiate most current MOEA algorithms is given in Table 1. The most important components are (i) the mating selection (**Mating**), and (ii) the environmental selection or truncation (**Replacement**).

Table 1. Algorithmic components of the MOEA framework used

Component	Domain	Description
μ	\mathbb{N}^+	Population size
μ_0	$[0, 1, 1] \subset \mathbb{R}$	Num. of initial solutions is $\mu \cdot \mu_0$
λ	\mathbb{N}^+	Number of offspring
pop	{ fixed-size, bounded }	Population type
pop _{ext}	{ none, bounded, unbounded }	External archive type
N_{ext}	\mathbb{N}^+	pop _{ext} size, if pop _{ext} type = <i>bounded</i>
Selection	{ random deterministic tournament stochastic tournament }	Mating selection operator
Removal	{ sequential one shot }	Population replacement policy
Set-partitioning	{ none dominance rank dominance strength dominance depth dominance depth-rank }	
Quality indicator	{ none binary indicator (ϵ or hypervolume) (IBEA) hypervolume contribution (SMS-EMOA) h-hypervolume contribution (HypE) }	
Diversity	{ none niche sharing k-th nearest neighbor crowding distance adaptive grid (PAES) }	

Traditionally, mating and replacement have been defined as compositions of a *fitness* component, designed to favor convergence, and a *diversity* component, meant to keep the population spread across the objective space. Following Zitzler et al. [23], we consider general preference relations comprising three lower-level components: (i) a *set-partitioning relation*, which partitions a set of solutions in a manner consistent with Pareto dominance but cannot discriminate between nondominated solutions; (ii) a *quality indicator*, which assigns a quality value to solutions in a manner that does not contradict Pareto dominance (often used as a refinement of the partitions); and (iii) a diversity metric, which does not need to be consistent with Pareto dominance. All the options implemented for these low-level components are shown in Table 1.

We define the **Mating** component as being composed of a preference relation and a selection method. The selection method may be any of the traditional methods in EAs: random, tournament, etc. Conversely, component **Replacement** is composed of a preference relation and a removal policy. The latter may be either *sequential* [15] (also called *iterative* [1]), i.e., one individual/solution is removed at a time and preference relations are re-computed before the next is discarded; or *one-shot*, i.e., preference relations are computed once and the worst solutions are removed altogether [1].

Table 2. Instantiation of MOEAs using our framework. Other components are parameters except $\text{pop} = \textit{fixed-size}$ and $\text{pop}_{\text{ext}} = \textit{none}$. SMS-EMOA uses $\lambda = 1$ by design.

Algorithm	Mating				Replacement			
	SetPart	Quality	Diversity	Selection	SetPart	Quality	Diversity	Removal
MOGA [10]	rank	—	sharing	det. tour.	rank	—	sharing	one-shot
NSGA-II [8]	depth	—	crowding	det. tour.	depth	—	crowding	one-shot
SPEA2 [22]	strength	—	k-NN	det. tour.	strength	—	k-NN	sequential
IBEA [21]	—	bin. ind.	—	det. tour.	—	bin. ind.	—	sequential
SMS-EMOA [2]	—	—	—	random	depth rank	hyperv. contrib.	—	—
HypE [1]	—	h -hyperv. contrib.	—	det. tour.	depth	h -hyperv. contrib.	—	sequential

Table 3. Parameter space for tuning the numerical parameters of all MOEAs

Parameter	μ	λ	pc	p_{mut}	p_X	Algorithm	IBEA	MOGA	SPEA2
Domain	{10, 20, ..., 100}	1 or $\lambda_r \cdot \mu$ $\lambda_r \in [0.1, 2]$	[0, 1]	[0, 1]	[0, 1]	Parameter	$indicator$	σ_{share}	k
						Domain	{ $I_c, I_{\bar{H}}$ }	[0.1, 1]	{1, ..., 9}

Besides the components explained above, populations and archives also need to be highlighted. Traditionally, a population is a set of individuals (dominated and nondominated alike) that are subject to the evolutionary process. As discussed elsewhere [19], an archive can be seen as a generalized population that may (i) keep only nondominated solutions and/or (ii) have unlimited capacity. In this work, we implement an internal archive pop (in place of the population), as well as an external archive pop_{ext} , which does not interfere with the evolutionary process. Concretely, if pop is set to *fixed-size*, it behaves like a regular population. If it is set to *bounded*, it behaves like an internal archive, accepting only nondominated solutions until its maximum capacity is reached. At this point, a replacement is carried out. By using this flexible implementation, we are able to instantiate algorithms such as SPEA2, which were originally described with an external archive that interferes in the evolutionary process. Concerning the external archive as implemented in this work, a MOEA may choose (i) not to use it ($\text{pop}_{\text{ext}} = \textit{none}$), (ii) to use it without capacity constraints ($\text{pop}_{\text{ext}} = \textit{unbounded}$), or (iii) to use it with a maximum capacity N_{ext} ($\text{pop}_{\text{ext}} = \textit{bounded}$). In the latter case, an additional replacement component $\text{Replacement}_{\text{Ext}}$ needs to be defined, similar to the replacement component used by the internal archive. Moreover, in the case of $\text{pop} = \textit{bounded}$, we add a numerical parameter μ_0 that determines the number of initial solutions as $\mu \cdot \mu_0$.

As shown in Table 2, by selecting the corresponding components for each algorithm, we can instantiate at least six well-known MOEAs from the literature. Particularly, SMS-EMOA and HypE use slightly different definitions of hypervolume contribution, and the definition used by HypE is further parametrized by a parameter h (k in the original paper [1]). SPEA2 uses a pre-defined formula for computing its k parameter value. Here, we implement both the formula and an option for manual input of k , i.e., k becomes a numerical parameter.

Table 4. The MOEAs are sorted according to their sum of ranks (in parenthesis)

C_{\max} TFT	AutoMOEA PFSP (249)	AutoMOEA C_{\max} -TFT (302)	IBEA (398)	NSGA-II (472)	SPEA2 (479)	HypE (585)	MOGA (687)	SMS (788)
C_{\max} TT	AutoMOEA C_{\max} -TT (209)	AutoMOEA PFSP (253)	NSGA-II (357)	SPEA2 (464)	IBEA (547)	HypE (574)	SMS (770)	MOGA (786)
TFT TT	MOGA (304)	IBEA (371)	AutoMOEA TFT-TT (475)	HypE (499)	NSGA-II (499)	AutoMOEA PFSP (553)	SPEA2 (615)	SMS (644)
C_{\max} TFT-TT	AutoMOEA (161)	AutoMOEA PFSP (251)	IBEA (417)	SPEA2 (525)	HypE (528)	NSGA-II (541)	SMS (735)	MOGA (802)

3 Experimental Setup

We consider the six standard MOEAs in Table 2 and we compare them with the novel, automatically designed MOEAs instantiated using this framework for the multi-objective PFSP variants. Our goal is to identify good combinations of such components, specially if they differ from (and are better than) existing combinations used by standard MOEAs.

We use *irace* [14] as tuner, adapting it to multi-objective optimization by using the hypervolume quality measure [16]. For computing the hypervolume, we normalize the objective values to the range $[1, 2]$ using the maximum and minimum ever found per problem, and use $(2.1, 2.1)$ and $(2.1, 2.1, 2.1)$, respectively, as reference points for the bi- and tri-objective variants. We set a budget of 5 000 runs for each tuning in order to design each variant-specific AutoMOEA. Since AutoMOEA_{PFSP} needs to see four times more instances than the variant-specific AutoMOEAs, we give it a tuning budget of 20 000 runs. Moreover, since the traditional MOEAs do not present default values for an application to the PFSP, we tune their numerical parameters using the same budget given to the variant-specific AutoMOEAs (5 000 runs per MOEA per variant per tuning). The parameter space used for the numerical parameters is the same for all MOEAs (Table 3). Following [5], all MOEAs use random initial solutions, unbounded external archives, two-point crossover and two problem-specific mutation operators, namely *insert* and *exchange*. In Table 3, p_{mut} stands for the probability of applying mutation to a given individual, while p_X is the probability of using the exchange operator if the first test is successful ($1 - p_X$ for the insertion operator).

To ensure that our results are not affected by overtuning, we use two independent benchmark sets for testing and tuning. The tuning benchmark set contains the instances with 20 machines from the tuning benchmark set proposed by [9]. The testing benchmark set is an adaptation of the benchmark set proposed by Taillard [18], following what is traditionally done in the multi-objective PFSP literature [9, 17]. Once the algorithms are tuned, we run them 10 times on each test instance, compute the average hypervolume, and compare the results using rank sum analysis and parallel coordinate plots. Experiments are run on a single core of Intel Xeon E5410 CPUs, running at 2.33GHz with 6MB of cache size under Cluster Rocks Linux version 6.0/CentOS 6.3. For brevity, few representative results are given here. The complete list of results is provided as supplementary material [4], as well as the parameter settings used by *irace*.

Table 5. Parameters selected by irace for AutoMOEA_{Cmax-TFT}, AutoMOEA_{Cmax-TT}, AutoMOEA_{TFT-TT}, AutoMOEA_{Cmax-TFT-TT}, and AutoMOEA_{PFSP}, respectively. All AutoMOEAs use pop = bounded.

Cmax-TFT		Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	λ_r	p_c	p_{mut}	p_x
Value	det. tour. (2)	rank	hyp. contr.	crowding	depth	bin. ind. (I_e)	crowding	one-shot	80	0.3	1.5	0.38	0.82	0.71	
Cmax-TT		Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	λ_r	p_c	p_{mut}	p_x
Value	random	—	—	—	—	h -hyp. contr.	crowding	one-shot	30	0.94	1.63	0.34	0.95	0.81	
TFT-TT		Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	λ_r	p_c	p_{mut}	p_x
Value	stoch. tour. (0.9)	—	—	crowding	count	bin. ind. (I_e)	sharing (0.87)	sequential	70	0.94	1.47	0.77	0.99	0.63	
Cmax-TFT-TT		Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	λ_r	p_c	p_{mut}	p_x
Value	det. tour. (2)	rank	—	crowding	—	h -hyp. contr.	crowding	one-shot	40	0.26	1.68	0.36	0.85	0.74	
PFSP		Mating				Replacement				Numerical					
Parameter	Selection	SetPart	Quality	Diversity	SetPart	Quality	Diversity	Removal	μ	μ_0	λ_r	λ_r	p_c	p_{mut}	p_x
Value	random	—	—	—	—	h -hyp. contr.	—	one-shot	60	0.73	1.53	0.17	0.76	0.40	

4 Results and Discussion

As shown in Table 5, the structures of $\text{AutoMOEA}_{\text{Cmax-TFT}}$ and $\text{AutoMOEA}_{\text{PFSP}}$ differ substantially. $\text{AutoMOEA}_{\text{Cmax-TFT}}$ combines components from almost all specific known MOEAs that built the basis for the algorithmic components included in the framework. Its mating preference relation joins components from MOGA (*dominance rank*), SMS-EMOA (*hypervolume contribution*), and NSGA-II (*crowding distance*). The replacement preference relation is a combination of the components from NSGA-II and IBEA. Interestingly, this very heterogeneous MOEA is able to outperform all traditional MOEAs on many of the tested instances. In fact, it is only outperformed by $\text{AutoMOEA}_{\text{PFSP}}$. The structure of $\text{AutoMOEA}_{\text{PFSP}}$ is radically simpler. The mating selection is performed randomly, so no preference relation is required. For replacement, only the shared hypervolume contribution proposed for HypE is used. However, the *one-shot* removal policy is adopted, most likely to cope with the computational overhead for the three-objective variant. Despite its relatively simple structure, $\text{AutoMOEA}_{\text{PFSP}}$ outperforms, in the Cmax-TFT problem variant, all MOEAs considered, including $\text{AutoMOEA}_{\text{Cmax-TFT}}$.

The similarity of the structures of $\text{AutoMOEA}_{\text{Cmax-TT}}$ and $\text{AutoMOEA}_{\text{PFSP}}$ is remarkable. In fact, the only significant differences between these two algorithms lie in the addition of the crowding distance diversity operator in $\text{AutoMOEA}_{\text{Cmax-TT}}$ and the population size, which in $\text{AutoMOEA}_{\text{Cmax-TT}}$ is half the size used by $\text{AutoMOEA}_{\text{PFSP}}$. These small differences are reflected on the similar rank sums they achieve. This time, however, the variant-specific $\text{AutoMOEA}_{\text{Cmax-TT}}$ outperforms the general $\text{AutoMOEA}_{\text{PFSP}}$ for several instances. Both algorithms obtain better hypervolume values for almost all test instances in comparison to the traditional MOEAs.

For the TFT-TT variant, results reflect the peculiarity of the problem’s structure. For this variant, the number of nondominated solutions decreases as the problem size increases. This is confirmed by the fact that many algorithms applied to this variant tend to find a single solution for the larger instances. Having this mixed nature, the design of $\text{AutoMOEA}_{\text{TFT-TT}}$ is unable to outperform some algorithms across the whole benchmark. In fact, MOGA is the only algorithm to perform well in the instances with few nondominated solutions. We suspect this unexpectedly good performance of MOGA can be explained by its capacity to establish and maintain niches, an effective strategy for single-objective EAs. Given the peculiarities of this variant, the high rank sum achieved by $\text{AutoMOEA}_{\text{PFSP}}$ confirms that using general-purpose designed algorithms may eventually present sub-optimal performance, and that using a variant-specific design one can overcome to some extent this type of drawbacks.

This mixed characteristics of this variant are shown in the parallel coordinate plots depicted in Fig. 1. For the “multi-objective” instances like the ones with 50 jobs and 20 machines (top), AutoMOEA often performs best. However, for the “single-objective” ones, like the ones with 200 jobs and 20 machines (bottom), all algorithms perform clearly worse than MOGA. To verify whether we could automatically generate an AutoMOEA as good as MOGA for this problem,

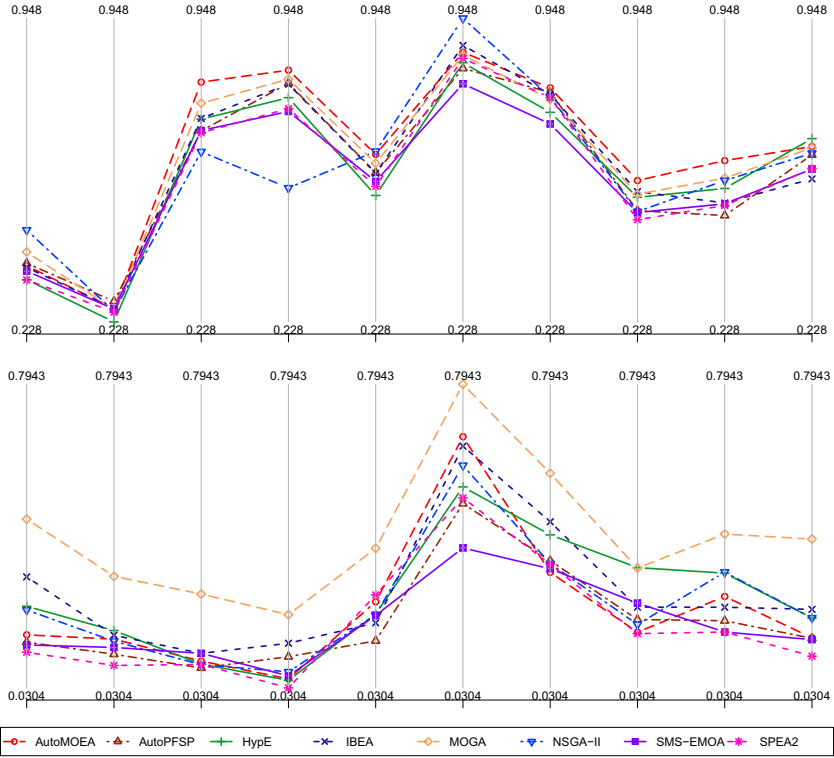


Fig. 1. MOEA’s average hypervolume on 10 instances of C_{\max} -TFT. Top: 50 jobs and 20 machines. Bottom: 200 jobs and 20 machines. Each vertical line depicts one instance.

we ran *irace* again under a different setup. First, we added the generational removal policy from MOGA, which we had excluded from the parameter space in an early development stage because of the evidence favoring elitism throughout the MO literature. Second, we added the configuration of MOGA as an initial candidate for *irace*. However, this candidate was discarded in the first iteration, and the final configuration was very similar to the one presented here. This could also be explained by possible differences in the training and test sets.

To conclude, the structure of $\text{AutoMOEA}_{C_{\max}\text{-TFT-IT}}$ combines elements from $\text{AutoMOEA}_{C_{\max}\text{-TFT}}$ (mating) and $\text{AutoMOEA}_{C_{\max}\text{-TT}}$ (replacement), which was expected since these components proved to be effective for these objectives. Interestingly, only the replacement hypervolume-based component is maintained. This is likely explained by the computational overhead that the hypervolume-based components introduce, particularly for three-objective scenarios. Concerning the performance of this AutoMOEA, the rank sum depicted in Table 4 shows that this AutoMOEA achieves hypervolumes that are better over almost all test instances considered. $\text{AutoMOEA}_{\text{PFSP}}$ ranks second, confirming that it is a flexible but effective algorithm.

5 Conclusions

In this paper, we have investigated the automatic design of multi-objective evolutionary algorithms (MOEAs). We used an extended framework that combines components from ParadisEO-MOEO, PISA, and PaGMO to deal with four different variants of the permutation flowshop problem (PFSP). We have used *irace*, an offline algorithm configuration tool, to automatically select the components that were more effective for each PFSP variant, thus producing novel MOEAs. These automatically designed MOEAs (AutoMOEAs) have outperformed the traditional MOEAs on all variants considered, confirming that novel combinations of existing MOEAs can lead to much improvement on the state-of-the-art, and reinforcing the need for research works on automatic algorithm design.

The analysis conducted here has reached its two main goals. The first was to assess whether the state of the art of MOEAs could be improved (and by how much) by using an automatic algorithm design methodology. The results shown here confirm that improvements can be achieved in the context of combinatorial optimization, particularly for the PFSP. The second was to investigate whether a single AutoMOEA could potentially outperform the variant-tailored AutoMOEAs, given that the variants present different search space characteristics. In terms of its structure, AutoMOEA_{PFSP} is a simple yet flexible and efficient algorithm. Concerning performance, the more general version outperforms the variant-specific AutoMOEA for the Cmax-TFT variant, but it is unable to match the performance of the other variant-specific AutoMOEAs. However, AutoMOEA_{PFSP} often outperforms the other MOEAs considered in this work.

Finally, the results presented here are preliminary. More experimental analysis, ranging from combinatorial to continuous, would be required to assess the full potential of the automatic design of MOEAs. Nonetheless, these initial results suggest that our approach is not only feasible but that it may give significant insights about the role various algorithmic components of MOEAs play and lead to completely new MOEA designs that have never been tested or considered previously. Moreover, deeper analysis on individual components and reasons for the high performance of the automatically designed configurations are an important open research question.

References

1. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation* 19(1), 45–76 (2011)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* 181(3), 1653–1669 (2007)
3. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Automatic generation of multi-objective ACO algorithms for the biobjective knapsack problem. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Engelbrecht, A.P., Groß, R., Stützle, T. (eds.) ANTS 2012. LNCS, vol. 7461, pp. 37–48. Springer, Heidelberg (2012)
4. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Automatic design of evolutionary algorithms for multi-objective combinatorial optimization: Supplementary material (2014), <http://iridia.ulb.ac.be/supp/IridiaSupp2014-007/>

5. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Deconstructing multi-objective evolutionary algorithms: An iterative analysis on the permutation flowshop. In: Pardalos, P., et al. (eds.) LION 8. LNCS. Springer (2014)
6. Biscani, F., Izzo, D., Yam, C.H.: A global optimisation toolbox for massively parallel engineering optimisation (2010), <http://arxiv.org/abs/1004.3824>
7. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – a platform and programming language independent interface for search algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6(2), 181–197 (2002)
9. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Comput. Oper. Res.* 38(8), 1219–1236 (2011)
10. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Forrest, S. (ed.) ICGA, pp. 416–423. Morgan Kaufmann Publishers (1993)
11. Igel, C., Heidrich-Meisner, V., Glasmachers, T.: Shark. *Journal of Machine Learning Research* 9, 993–996 (2008)
12. Knowles, J.D., Corne, D.: Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation* 8(2), 149–172 (2000)
13. Liefoghe, A., Jourdan, L., Talbi, E.G.: A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. *Eur. J. Oper. Res.* 209(2), 104–112 (2011)
14. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)
15. López-Ibáñez, M., Knowles, J., Laumanns, M.: On sequential online archiving of objective vectors. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 46–60. Springer, Heidelberg (2011)
16. López-Ibáñez, M., Stützle, T.: The automatic design of multi-objective ant colony optimization algorithms. *IEEE Trans. Evol. Comput.* 16(6), 861–875 (2012)
17. Minella, G., Ruiz, R., Ciavotta, M.: A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing* 20(3), 451–471 (2008)
18. Taillard, É.D.: Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* 64(2), 278–285 (1993)
19. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation* 8(2), 125–147 (2000)
20. Zhang, Q., Suganthan, P.N.: Special session on performance assessment of multi-objective optimization algorithms/CEC 2009 MOEA competition (2009)
21. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
22. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., et al. (eds.) EUROGEN 2001, pp. 95–100. CIMNE, Barcelona (2002)
23. Zitzler, E., Thiele, L., Bader, J.: On set-based multiobjective optimization. *IEEE Trans. Evol. Comput.* 14(1), 58–79 (2010)
24. Zitzler, E., Thiele, L., Deb, K.: Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* 8(2), 173–195 (2000)

Generic Postprocessing via Subset Selection for Hypervolume and Epsilon-Indicator[★]

Karl Bringmann¹, Tobias Friedrich², and Patrick Klitzke³

¹ Max Planck Institute for Informatics, Saarbrücken, Germany

² Friedrich-Schiller-Universität Jena, Jena, Germany

³ Universität des Saarlandes, Saarbrücken, Germany

Abstract. Most biobjective evolutionary algorithms maintain a population of fixed size μ and return the final population at termination. During the optimization process many solutions are considered, but most are discarded. We present two generic postprocessing algorithms which utilize the archive of all non-dominated solutions evaluated during the search. We choose the best μ solutions from the archive such that the hypervolume or ε -indicator is maximized. This postprocessing costs no additional fitness function evaluations and has negligible runtime compared to most EMOAs.

We experimentally examine our postprocessing for four standard algorithms (NSGA-II, SPEA2, SMS-EMOA, IBEA) on ten standard test functions (DTLZ 1–2,7, ZDT 1–3, WFG 3–6) and measure the average quality improvement. The median decrease of the distance to the optimal ε -indicator is 95%, the median decrease of the distance to the optimal hypervolume value is 86%. We observe similar performance on a real-world problem (wind turbine placement).

1 Introduction

Biobjective optimization aims at minimizing (or maximizing) a two-dimensional fitness function $f: \mathcal{X} \rightarrow \mathbb{R}^2$. As the two objectives f_1 and f_2 are typically contradicting, the outcome of the optimization is a set of incomparable solutions describing a Pareto front. Multiobjective evolutionary algorithms (MOEA) typically maintain a set of solutions called population during the optimization. The simplest MOEAs (like SEMO [13, 15, 16]) keep all non-dominated solutions in the population. As the Pareto front of a biobjective fitness function can be exponential in the input size [11], this results in exponential runtimes of SEMO for such fitness functions [7, 14]. More advanced MOEAs therefore avoid keeping all non-dominated solutions in the population and assume some upper limit on the size of the population.

[★] The research leading to these results has received funding from the Australian Research Council (ARC) under grant agreement DP140103400 and from the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement no 618091 (SAGE). K.B. is a recipient of the *Google Europe Fellowship in Randomized Algorithms*, and this research is supported in part by this Google Fellowship.

With a population of fixed maximum size, MOEAs have to decide in each step which solutions to keep in the population and which solutions to remove. This can be done based on various measures like the hypervolume contribution [24], ε -approximation, crowding distance, or many others. Independent of the specific measure, the archiving algorithm of a MOEA has to solve an online problem: It has to decide which solutions to remove without knowing what new solutions will be generated in the future. While the MOEA has to decide ‘online’ which solutions to keep, an ‘offline’ algorithm would have access to all points generated during the optimization process and would just choose the best set from this archive. It is known that a MOEA which in each iteration keeps these μ solutions in the population that maximize the hypervolume, can still only reach a final hypervolume which is a factor μ smaller than achieved by the optimal choice of μ solutions from the whole archive (in the worst case) [3].

This shows that MOEAs potentially lose a lot of information by dropping solutions during the optimization. We suggest to make best use of the accumulated information by collecting all search points seen during the optimization in an archive. After the MOEA has stopped, we suggest to do a postprocessing that selects the best subset (of size μ) from the whole archive. This costs no additional fitness evaluations and should not reduce the quality of the reported final population. Depending on the ultimate aim of the optimization, the problem solved by this postprocessing is known as the *Hypervolume Subset Selection Problem* (HYPSSP) [1, 6, 17] or *ε -Indicator Subset Selection Problem* (EPS SSP) [6, 19, 21]. Until recently the best known runtimes for these subset selection problems were quadratic in the archive size n . As a single run of a MOEA can easily produce $n = 10^5$ or more non-dominated search points, a runtime of $\mathcal{O}(n^2)$ is prohibitively large. The postprocessing therefore only becomes tractable due to two recently published quasi-linear algorithms by the authors [6], specifically, an algorithm for HYPSSP with runtime $\mathcal{O}(n(\mu + \log n))$ and one for EPS SSP with runtime $\mathcal{O}(n \log n)$.

In this paper, we are going to investigate the effect of these two postprocessing algorithms for four standard algorithms on 10 common multi-objective optimization test functions and one real-world problem of optimizing the placement of wind turbines [20, 22].

Quality Measures. There are several ways to measure the quality of solution sets. We focus on two measures. Our first metric is the hypervolume indicator. It measures the volume of the objective space dominated by the set of solutions relative to a reference point. Its main disadvantage is its high computational complexity in higher dimensions [2, 4]. Our second metric is the ε -indicator. For measuring how well a set P approximates another set R , the (additive) ε -indicator returns the minimal ε by which we have to increase all points in P in all coordinates so that every point in R is dominated by some point in P . The disadvantage of this notion is that its computation requires knowing the Pareto front, as we would like to plug in the Pareto front for R . In contrast to the hypervolume indicator, it can therefore not directly be used to guide the search [5].

2 Preliminaries

We consider biobjective minimization problems, where a vector-valued function $f = (f_1, f_2): \mathcal{X} \rightarrow \mathbb{R}^2$ is minimized with respect to the weak Pareto dominance relation \preceq . We will mainly work in the *objective space* $f(\mathcal{X})$ and say that a point $p = (p_1, p_2) \in \mathbb{R}^2$ *weakly dominates* another point $q = (q_1, q_2) \in \mathbb{R}^2$ (denoted as $p \preceq q$) iff $p_1 \leq q_1$ and $p_2 \leq q_2$. The aim of most MOEAs is to approximate the *Pareto front* $\mathcal{F} = \{f(x) \mid x \in \mathcal{X}: \nexists y \in \mathcal{X}: f(y) \preceq f(x) \wedge f(x) \not\preceq f(y)\}$.

Hypervolume Indicator. For a set of points $P \subset \mathbb{R}^2$, the *hypervolume indicator* is defined as the volume of the set of points that are weakly dominated by solutions in P and at the same time weakly dominate a given reference point $r \in \mathbb{R}^2$, that is, $\mathcal{I}_{\text{hyp}}(P) := \mathcal{I}_{\text{hyp}}(P, r) := \lambda(\{z \in \mathbb{R}^2 \mid \exists a \in P: a \preceq z \preceq r\})$, where λ is the Lebesgue measure.

The *Hypervolume Subset Selection Problem* (HYPSSP) is then defined as follows: Given a set $P \subset \mathbb{R}^2$ of size n , $r \in \mathbb{R}^2$, and $\mu \in \mathbb{N}$, compute a subset $P^* \subseteq P$ of size at most μ that maximizes $\mathcal{I}_{\text{hyp}}(P^*, r)$. We write the result of this problem as $\text{HYPSSP}(P, r, \mu)$.

ε -Indicator. How well a point $p = (p_1, p_2) \in \mathbb{R}^2$ approximates another point $r = (r_1, r_2) \in \mathbb{R}^2$ in the objective space can be measured by the minimal number ε by which we have to decrease p in both coordinates so that it dominates r . More formally, we set $\mathcal{I}_{\text{eps}}(p, r) := \max\{p_1 - r_1, p_2 - r_2\}$. This can be used to define how well a set $P \subset \mathbb{R}^2$ approximates a set $R \subset \mathbb{R}^2$: The ε -indicator is defined as $\mathcal{I}_{\text{eps}}(P, R) := \max_{r \in R} \min_{p \in P} \mathcal{I}_{\text{eps}}(p, r)$. This denotes the minimal number ε by which we have to decrease all points in P in both coordinates so that every point in R is dominated by some point in P .

The *ε -Indicator Subset Selection Problem* (EPS SSP) is defined as follows: Given a set $P \subset \mathbb{R}^d$ of size n , $R \subset \mathbb{R}^d$ of size m , and $\mu \in \mathbb{N}$, compute a subset $P^* \subseteq P$ of size at most μ that minimizes $\mathcal{I}_{\text{eps}}(P^*, R)$. We write the result of this problem as $\text{EPS SSP}(P, R, \mu)$.

3 Postprocessing

Consider any EMOA with population size μ running until it performed n fitness evaluations. Let P be the final population after n fitness evaluations and A be the archive of all n solutions that were evaluated during the run. We describe our postprocessing in the objective space, i.e., we let $P, A \subset \mathbb{R}^2$.

Hypervolume. For \mathcal{I}_{hyp} we may pick any reference point $r \in \mathbb{R}^2$. The general optimization goal is then to find a population P^* of μ Pareto optimal points that maximize the hypervolume, i.e., $P^* = \text{HYPSSP}(\mathcal{F}, r, \mu)$. Unfortunately, the Pareto front is unknown. To overcome this problem, we introduce the assumption that the archive converges to the Pareto front (as has been done in AGE [5] and is implicit in the design of most EMOAs). Thus, our \mathcal{I}_{hyp} -postprocessing computes the set of μ points maximizing the hypervolume among all points in the archive,

$$\text{PP}_{\text{hyp}}(A, \mu) := \text{HYPSSP}(A, r, \mu).$$

Typically the hypervolume of $\text{PP}_{\text{hyp}}(A, \mu)$ should be larger than the hypervolume of P , so that our postprocessing improves the quality. In Section 5 we will see an experimental evaluation of this claim. In any case, since $P \subseteq A$ we have $\mathcal{I}_{\text{hyp}}(\text{PP}_{\text{hyp}}(A, \mu)) \geq \mathcal{I}_{\text{hyp}}(P)$, so our postprocessing does not decrease the quality of the result.

ε -Indicator. In case of \mathcal{I}_{eps} the general optimization goal is to find a population P^* of μ Pareto optimal points that optimally approximate the Pareto front, i.e., $P^* = \text{EPS SSP}(\mathcal{F}, \mathcal{F}, \mu)$. Again, \mathcal{F} is unknown and we assume that the archive converges to the Pareto front. Thus, our \mathcal{I}_{eps} -postprocessing computes the set of μ points among all points in the archive that best approximate the archive,

$$\text{PP}_{\text{eps}}(A, \mu) := \text{EPS SSP}'(A, A, \mu).$$

Here, the prime in $\text{EPS SSP}'$ hides a minor modification that improves the experimental results, namely that we choose a population P among all μ -subsets of A that include the leftmost point of A and the bottommost point of A , i.e., the single-objective optima. Intuitively, this is necessary since these extrema are needed to cover the boundaries of the Pareto front. We remark that one can compute $\text{EPS SSP}'(A, A, \mu)$ with a minor modification of [6].

Again, typically $\mathcal{I}_{\text{eps}}(\text{PP}_{\text{eps}}(A, \mu), \mathcal{F})$ should be smaller than $\mathcal{I}_{\text{eps}}(P, \mathcal{F})$, so that our postprocessing improves the quality, and we will examine this claim experimentally. A noteworthy difference to the hypervolume case is that our postprocessing for \mathcal{I}_{eps} does not come with the guarantee that quality cannot deteriorate, in fact, we will see in Section 5 that worsenings can happen but are rare.

Complexity. For a population size of μ and n fitness evaluations, NSGA-II, SPEA2, and IBEA have a running time of $\mathcal{O}(n\mu \log \mu)$, while SMS-EMOA has a running time of $\mathcal{O}(n\mu^2)$. Our postprocessing takes time $\mathcal{O}(n(\mu + \log n))$ for \mathcal{I}_{hyp} and $\mathcal{O}(n \log n)$ for \mathcal{I}_{eps} [6], which is comparable to the runtime of typical EMOAs. In our experiments, the postprocessing tends to be even faster, since we only have to store the non-dominated points of the archive, which are much less than n .

4 Experimental Setup

Implementation and Hardware. All presented algorithms have been implemented in Java using the jMetal framework [10] and run on a compute cluster with 128 nodes, each having two Intel Xeon E5620 @ 2.40GHz. The code is available at <http://docs.theinf.uni-jena.de/code/ssp.zip>.

Benchmark Problems and EMOAs. We compared the improvement gained by the postprocessing for the well established EMOAs NSGA-II [8], SPEA2 [26], SMS-EMOA [12], and IBEA [23]. As test functions we used DTLZ1, DTLZ2, DTLZ7 from DTLZ [9] with 7 variables, ZDT1–3 from ZDT [25] with 30 variables, and WFG3–6 from the [18] with 4 variables. We chose these benchmarks, because explicit expressions for their Pareto fronts are readily available. For measuring the hypervolume, we choose the reference point $r = (11, 11)$.

Additionally to the standard test functions, we used a simulation of a wind turbine placement function [20], which optimizes for the maximum power and the minimum perimeter of the convex hull formed by the turbine positions with 30 turbines on a discrete area of size 3000×3000 .

All experimental results (medians, quartiles, ...) that we will report in the next section are based on 700 independent runs for the benchmark problems and 100 independent runs for the turbine. As population size we used 100 for the benchmark problems and 10 for the turbine.

Quality Evaluation. We want to compare the quality $\mathcal{I}_{\text{hyp}}(P, r)$ of a population P with the optimal hypervolume $\text{OPT}_{\text{hyp}} := \text{HYPSSP}(\mathcal{F}, \mu)$ of any μ points on the Pareto front. For measuring the proximity to the front, we measure $\mathcal{I}_{\text{hyp}}^{\Delta}(P, r) := \text{OPT}_{\text{hyp}} - \mathcal{I}_{\text{hyp}}(P, r)$ in our experiments. However, as \mathcal{F} is infinite it seems impossible to compute OPT_{hyp} . Instead of \mathcal{F} we therefore consider a set $\mathcal{F}' \subseteq \mathcal{F}$ of m points placed equidistantly along the front, which we can compute because we know an explicit expression for \mathcal{F} for the chosen benchmark problems. Now we simply replace \mathcal{F} by \mathcal{F}' in the definition of OPT_{hyp} to obtain an approximation. In case of \mathcal{I}_{eps} , for the same reasons we cannot directly compute $\text{OPT}_{\text{eps}} := \text{EPS SSP}(\mathcal{F}, \mathcal{F}, \mu)$. Moreover, we have the additional difficulty that we cannot evaluate the quality $\mathcal{I}_{\text{eps}}(P, \mathcal{F})$ of a population P . Again, we replace \mathcal{F} by its finite approximation \mathcal{F}' and obtain approximations for the optimum and the quality of a population. We use $\mathcal{I}_{\text{eps}}^{\Delta}(P, \mathcal{F}) := \text{OPT}_{\text{eps}} - \mathcal{I}_{\text{eps}}(P, \mathcal{F})$ as quality measure.

In our experiments we choose $m = 10^6$, which makes the error smaller than any of our reported values, and we ignore this error from now on. Note that it is infeasible to make m much larger, since the runtime for computing the approximation of, e.g., OPT_{hyp} is $\mathcal{O}(m(\mu + \log m))$.

Statistics. Additionally to calculating the (median) quality with and without postprocessing, we also perform a non-parametric test on the significance of the observed behavior. For this, we use the Wilcoxon-Mann-Whitney two-sample rank-sum test at the 95% confidence level.

5 Experimental Results

Test functions. The results of our experimental study on standard test functions are presented in Figure 1. The tables show the median of the indicators $\mathcal{I}_{\text{hyp}}^{\Delta}$ and $\mathcal{I}_{\text{eps}}^{\Delta}$ after 100 000 fitness evaluations. Our postprocessing improves (or does not worsen) the hypervolume and ε -indicator for all functions and algorithms. In all but six (out of 80 combinations) the improvement is statistically significant at the 95% confidence level. In fact, the median hypervolume is always increased and the distance to the optimal hypervolume $\mathcal{I}_{\text{hyp}}^{\Delta}$ therefore decreased. In 37 out of 40 cases the median $\mathcal{I}_{\text{hyp}}^{\Delta}$ decreases by more than 0.01%. The median reduction of $\mathcal{I}_{\text{hyp}}^{\Delta}$ for all 40 combinations of algorithms and functions is -85.6% (mean -60.5%). On the other hand, the median $\mathcal{I}_{\text{eps}}^{\Delta}$ could be decreased by more than 0.01% in 36 out of 40 cases. The median reduction of $\mathcal{I}_{\text{eps}}^{\Delta}$ is -95.2% (mean -73.4%).

Function	NSGA-II	IBEA	SPEA2	SMS-EMOA
DTLZ1	$4.7 \cdot 10^{-4} \rightarrow 7.1 \cdot 10^{-5}$ (-85.0%)	$8.1 \cdot 10^{-2} \rightarrow 3.1 \cdot 10^{-4}$ (-99.6%)	$2.0 \cdot 10^{-1} \rightarrow 5.4 \cdot 10^{-5}$ (-99.9%)	$3.1 \cdot 10^{-5} \rightarrow 2.8 \cdot 10^{-5}$ (-10.1%)
DTLZ2	$1.9 \cdot 10^{-3} \rightarrow 1.3 \cdot 10^{-5}$ (-99.3%)	$1.3 \cdot 10^{-2} \rightarrow 1.1 \cdot 10^{-5}$ (-99.9%)	$1.7 \cdot 10^{-1} \rightarrow 1.2 \cdot 10^{-5}$ (-99.9%)	$1.9 \cdot 10^{-5} \rightarrow 1.1 \cdot 10^{-5}$ (-41.3%)
DTLZ7	$2.1 \cdot 10 \rightarrow 2.1 \cdot 10$ ($\pm 0\%$)	$3.4 \cdot 10 \rightarrow 3.4 \cdot 10$ (-0.04%)	$2.2 \cdot 10 \rightarrow 2.1 \cdot 10$ (-0.6%)	$2.1 \cdot 10 \rightarrow 2.1 \cdot 10$ ($\pm 0\%$)
ZDT1	$1.9 \cdot 10^{-3} \rightarrow 1.8 \cdot 10^{-5}$ (-99.1%)	$5.4 \cdot 10^{-2} \rightarrow 8.2 \cdot 10^{-5}$ (-99.8%)	$6.0 \cdot 10^{-2} \rightarrow 2.3 \cdot 10^{-5}$ (-99.9%)	$2.7 \cdot 10^{-5} \rightarrow 1.4 \cdot 10^{-5}$ (-46.9%)
ZDT2	$1.7 \cdot 10^{-3} \rightarrow 1.4 \cdot 10^{-5}$ (-99.2%)	$1.6 \cdot 10^{-2} \rightarrow 1.1 \cdot 10^{-5}$ (-99.9%)	$1.8 \cdot 10^{-1} \rightarrow 2.9 \cdot 10^{-5}$ (-99.9%)	$3.7 \cdot 10^{-5} \rightarrow 1.9 \cdot 10^{-5}$ (-47.3%)
ZDT3	$1.1 \cdot 10^{-3} \rightarrow 6.1 \cdot 10^{-6}$ (-99.4%)	$3.5 \cdot 10^{-2} \rightarrow 3.3 \cdot 10^{-5}$ (-99.9%)	$5.7 \cdot 10^{-2} \rightarrow 9.2 \cdot 10^{-6}$ (-99.9%)	$1.3 \cdot 10^{-5} \rightarrow 5.1 \cdot 10^{-6}$ (-59.3%)
WFG3	$2.9 \cdot 10^{-2} \rightarrow 1.2 \cdot 10^{-2}$ (-60.1%)	$2.9 \cdot 10^{-1} \rightarrow 7.7 \cdot 10^{-3}$ (-97.4%)	$8.1 \cdot 10^{-1} \rightarrow 1.2 \cdot 10^{-2}$ (-98.5%)	$7.3 \cdot 10^{-3} \rightarrow 7.2 \cdot 10^{-3}$ (-2.1%)
WFG4	$1.8 \cdot 10^{-2} \rightarrow 1.7 \cdot 10^{-3}$ (-90.5%)	$9.5 \cdot 10^{-2} \rightarrow 3.8 \cdot 10^{-5}$ (-99.9%)	1.2 $\rightarrow 1.2 \cdot 10^{-2}$ (-99.0%)	$1.2 \cdot 10^{-2} \rightarrow 1.2 \cdot 10^{-2}$ (-0.4%)
WFG5	1.7 $\rightarrow 1.6$ (-0.8%)	1.7 $\rightarrow 1.6$ (-2.7%)	2.8 $\rightarrow 1.7$ (-40.3%)	1.8 $\rightarrow 1.8$ ($\pm 0\%$)
WFG6	$1.9 \cdot 10^{-1} \rightarrow 1.8 \cdot 10^{-1}$ (-7.2%)	$4.0 \cdot 10^{-1} \rightarrow 2.0 \cdot 10^{-1}$ (-49.0%)	1.3 $\rightarrow 1.9 \cdot 10^{-1}$ (-86.2%)	$2.3 \cdot 10^{-1} \rightarrow 2.3 \cdot 10^{-1}$ (-0.03%)

(a) Distance to optimal hypervolume: $\mathcal{I}_{\text{hyp}}^{\Delta}(P) = \text{OPT}_{\text{hyp}} - \mathcal{I}_{\text{hyp}}(P)$.

Function	NSGA-II	IBEA	SPEA2	SMS-EMOA
DTLZ1	$4.1 \cdot 10^{-3} \rightarrow 2.3 \cdot 10^{-4}$ (-94.3%)	$2.0 \cdot 10^{-1} \rightarrow 7.5 \cdot 10^{-3}$ (-96.2%)	$2.5 \cdot 10^{-2} \rightarrow 2.2 \cdot 10^{-4}$ (-99.1%)	$3.5 \cdot 10^{-4} \rightarrow 1.7 \cdot 10^{-4}$ (-50.4%)
DTLZ2	$8.2 \cdot 10^{-3} \rightarrow 1.7 \cdot 10^{-4}$ (-97.9%)	$4.0 \cdot 10^{-2} \rightarrow 1.2 \cdot 10^{-4}$ (-99.7%)	$2.4 \cdot 10^{-2} \rightarrow 1.4 \cdot 10^{-4}$ (-99.4%)	$6.7 \cdot 10^{-4} \rightarrow 1.2 \cdot 10^{-4}$ (-82.3%)
DTLZ7	2.3 $\rightarrow 2.3$ ($\pm 0\%$)	3.6 $\rightarrow 3.6$ (-0.03%)	2.3 $\rightarrow 2.3$ ($\pm 0\%$)	2.3 $\rightarrow 2.3$ ($\pm 0\%$)
ZDT1	$8.3 \cdot 10^{-3} \rightarrow 1.7 \cdot 10^{-4}$ (-98.0%)	$4.0 \cdot 10^{-2} \rightarrow 1.0 \cdot 10^{-4}$ (-99.7%)	$2.2 \cdot 10^{-2} \rightarrow 1.5 \cdot 10^{-4}$ (-99.3%)	$5.7 \cdot 10^{-4} \rightarrow 1.2 \cdot 10^{-4}$ (-79.7%)
ZDT2	$6.1 \cdot 10^{-3} \rightarrow 1.5 \cdot 10^{-4}$ (-98.1%)	$4.1 \cdot 10^{-2} \rightarrow 1.8 \cdot 10^{-4}$ (-99.6%)	$2.5 \cdot 10^{-2} \rightarrow 1.5 \cdot 10^{-4}$ (-99.4%)	$5.5 \cdot 10^{-4} \rightarrow 1.3 \cdot 10^{-4}$ (-76.1%)
ZDT3	$8.0 \cdot 10^{-3} \rightarrow 1.0 \cdot 10^{-4}$ (-98.3%)	$2.4 \cdot 10^{-2} \rightarrow 3.8 \cdot 10^{-4}$ (-98.4%)	$2.5 \cdot 10^{-2} \rightarrow 1.2 \cdot 10^{-4}$ (-99.5%)	$1.3 \cdot 10^{-3} \rightarrow 6.6 \cdot 10^{-5}$ (-94.9%)
WFG3	$2.5 \cdot 10^{-2} \rightarrow 1.1 \cdot 10^{-3}$ (-95.5%)	$1.1 \cdot 10^{-1} \rightarrow 6.5 \cdot 10^{-4}$ (-99.4%)	$8.3 \cdot 10^{-2} \rightarrow 1.1 \cdot 10^{-3}$ (-98.7%)	$1.9 \cdot 10^{-3} \rightarrow 6.2 \cdot 10^{-4}$ (-66.7%)
WFG4	$2.4 \cdot 10^{-2} \rightarrow 5.2 \cdot 10^{-4}$ (-97.8%)	$8.7 \cdot 10^{-2} \rightarrow 2.3 \cdot 10^{-4}$ (-99.7%)	$1.0 \cdot 10^{-1} \rightarrow 5.0 \cdot 10^{-4}$ (-99.5%)	$2.2 \cdot 10^{-3} \rightarrow 1.7 \cdot 10^{-4}$ (-92.4%)
WFG5	$1.2 \cdot 10^{-1} \rightarrow 1.2 \cdot 10^{-1}$ ($\pm 0\%$)	$1.3 \cdot 10^{-1} \rightarrow 1.2 \cdot 10^{-1}$ (-10.3%)	$2.1 \cdot 10^{-1} \rightarrow 1.2 \cdot 10^{-1}$ (-41.0%)	$1.4 \cdot 10^{-1} \rightarrow 1.4 \cdot 10^{-1}$ ($\pm 0\%$)
WFG6	$3.3 \cdot 10^{-2} \rightarrow 8.7 \cdot 10^{-3}$ (-73.8%)	$1.2 \cdot 10^{-1} \rightarrow 9.8 \cdot 10^{-3}$ (-91.9%)	$1.1 \cdot 10^{-1} \rightarrow 8.7 \cdot 10^{-3}$ (-92.3%)	$1.2 \cdot 10^{-2} \rightarrow 1.0 \cdot 10^{-2}$ (-16.2%)

(b) Distance to optimal ε -Indicator: $\mathcal{I}_{\text{eps}}^{\Delta}(P) = \text{OPT}_{\text{eps}} - \mathcal{I}_{\text{eps}}(P, \mathcal{F})$.

Fig. 1. Medians after 100 000 evaluations. We show the value of the respective indicator before and after the postprocessing. We also give the factor by which the indicator got smaller (=better). Improvements of less than 0.01% are treated as equal and marked $\pm 0\%$. For each test function we marked the best indicator values before and after postprocessing in blue.

Figure 2 gives a more detailed view on how the indicators $\mathcal{I}_{\text{hyp}}^{\Delta}$ and \mathcal{I}_{eps} decrease over time for some exemplary combinations of MOEAs and test functions. Shown are box-and-whisker plots that specify the median, quartiles, and whiskers from minimum to maximum. Figures 2a and 2b show the typical behaviour of our postprocessing for NSGA-II. It is interesting to compare Figure 2b with Figure 3a, which shows the size of the population and of the archive over time. This shows that our postprocessing starts to “kick in” at the time we are starting to throw away points, where the archive size diverges from the population size.

Figure 2c shows the visually largest improvement that our \mathcal{I}_{hyp} -postprocessing was able to achieve for any benchmark problem and MOEA. Note that SPEA2 without postprocessing converges to a suboptimal hypervolume, but our \mathcal{I}_{hyp} -postprocessing is able to correct this. Figure 2d shows a similar profitable situation for \mathcal{I}_{eps} -postprocessing.

Since SMS-EMOA is hypervolume driven, one expects that \mathcal{I}_{hyp} -postprocessing is less effective for this algorithm compared to other EMOAs. Indeed, Figure 1 confirms this expectation, specifically, the median improvement of $\mathcal{I}_{\text{hyp}}^{\Delta}$ for SMS-EMOA is only -6.1% (mean -20.8%). However, Figure 2e shows that even the hypervolume driven SMS-EMOA can benefit a lot from our \mathcal{I}_{hyp} -postprocessing in some situations.

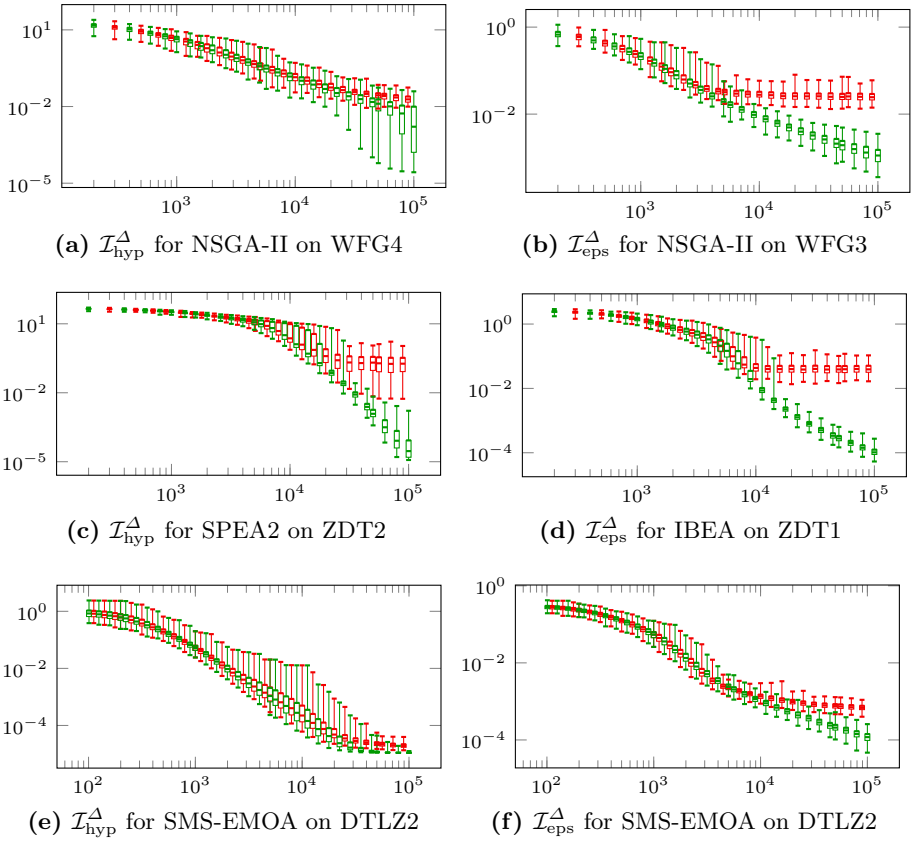


Fig. 2. Quality measures as a function of time (evaluations) before (in red) and after (in green) postprocessing for some exemplary combinations of MOEAs and test functions.

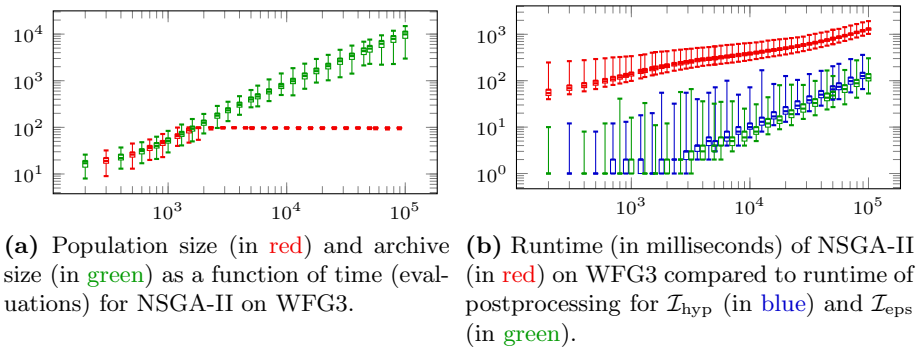


Fig. 3. Population size and runtime of NSGA-II on WFG3

In contrast to the \mathcal{I}_{hyp} -postprocessing, the \mathcal{I}_{eps} -postprocessing can (at least theoretically) make \mathcal{I}_{eps} worse. However, Figure 2f shows the only case where we observed that the \mathcal{I}_{eps} -postprocessing visually worsens the algorithm at some time, namely in a thin region from 2500 to 5000 fitness evaluations. After that point, the \mathcal{I}_{eps} -postprocessing gives again a huge improvement.

Figure 3b shows the total runtime up to n fitness evaluations of NSGA-II, the runtime of the other algorithms is similar (with SMS-EMOA being somewhat slower). Moreover, the runtime of the postprocessing when started after n fitness evaluations is plotted. This shows that the runtime of both postprocessing algorithms is negligible compared to the runtime of the MOEA.

Additionally, we examined whether a hypervolume-based algorithm (like SMS-EMOA) achieves more hypervolume than a non-hypervolume-based algorithms (like NSGA-II, SPEA2) with \mathcal{I}_{hyp} -postprocessing. To this end, we compared SMS-EMOA without postprocessing to the other three algorithms with postprocessing. The Wilcoxon-Mann-Whitney U-test at 95% confidence level showed that NSGA-II with \mathcal{I}_{hyp} -postprocessing outperforms SMS-EMOA without postprocessing on 7 out of 10 test functions. The same holds for SPEA2, but not for IBEA. This shows that our \mathcal{I}_{hyp} -postprocessing makes algorithms that do not aim at maximizing \mathcal{I}_{hyp} very competitive, even compared to algorithms that directly optimize \mathcal{I}_{hyp} . As one might expect, we no longer observe this behavior if SMS-EMOA is also allowed postprocessing: SMS-EMOA with \mathcal{I}_{hyp} -postprocessing achieves higher hypervolumes than all other algorithms with \mathcal{I}_{hyp} -postprocessing (on more than half of the test functions).

Figure 4 shows an exemplary result for the turbine problem for SMS-EMOA. Note that we cannot plot $\mathcal{I}_{\text{hyp}}^\Delta$ as OPT_{hyp} is not known. We observe a significant improvement also for this real-world problem. Specifically, after 300 000 evaluations our \mathcal{I}_{hyp} -postprocessing increased \mathcal{I}_{hyp} by 14%.

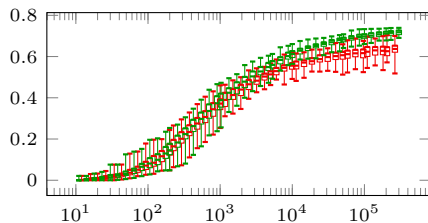


Fig. 4. Experimental results for turbine: \mathcal{I}_{hyp} (not $\mathcal{I}_{\text{hyp}}^\Delta$!) over time (evaluations) for SMS-EMOA before (in red) and after (in green) postprocessing

6 Conclusion

We studied two generic postprocessing methods which have the potential to improve the final output of any EMOA on any biobjective optimization problem. These methods choose the optimal subset of μ solutions from the archive of all solutions seen during the run of an EMOA such that the hypervolume or ε -indicator is optimized. This requires no additional fitness evaluations and therefore zero additional ‘optimization time’. Moreover, the computation time of our postprocessing methods is negligible compared to the computation time of typical EMOAs. We experimentally evaluated the quality of our postprocessing

on four standard EMOAs and ten standard test functions and one real-world problem. This showed that our postprocessing typically returns a set of solutions which is about 90% closer to the optimum than the regular outcome of the EMOAs.

References

- [1] Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In: Genetic and Evolutionary Computation Conference, GECCO 2009, pp. 563–570 (2009)
- [2] Bringmann, K., Friedrich, T.: Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry: Theory and Applications* 43, 601–610 (2010)
- [3] Bringmann, K., Friedrich, T.: Convergence of hypervolume-based archiving algorithms II: Competitiveness. In: Genetic and Evolutionary Computation Conference, GECCO 2012, pp. 457–464 (2012)
- [4] Bringmann, K., Friedrich, T.: Parameterized average-case complexity of the hypervolume indicator. In: Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 575–582 (2013)
- [5] Bringmann, K., Friedrich, T., Neumann, F., Wagner, M.: Approximation-guided evolutionary multi-objective optimization. In: 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011, pp. 1198–1203. IJCAI/AAAI (2011)
- [6] Bringmann, K., Friedrich, T., Klitzke, P.: Two-dimensional subset selection for hypervolume and epsilon-indicator. In: Genetic and Evolutionary Computation Conference, GECCO (2014)
- [7] Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: On the effects of adding objectives to plateau functions. *IEEE Trans. Evolutionary Computation* 13(3), 591–603 (2009)
- [8] Deb, K., Pratap, A., Agrawal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation* 6(2), 182–197 (2002)
- [9] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: *Evolutionary Multiobjective Optimization, Advanced Information and Knowledge Processing*, pp. 105–145 (2005)
- [10] Durillo, J.J., Nebro, A.J., Alba, E.: The jMetal framework for multi-objective optimization: Design and architecture. In: *IEEE Congress on Evolutionary Computation, CEC 2010*, pp. 4138–4325 (2010)
- [11] Ehrgott, M.: *Multicriteria Optimization*, 2nd edn. Springer (2005)
- [12] Emmerich, M.T.M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: *3rd International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005*, pp. 62–76 (2005)
- [13] Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation* 18(4), 617–633 (2010)
- [14] Friedrich, T., Hebbinghaus, N., Neumann, F.: Plateaus can be harder in multi-objective optimization. *Theoretical Computer Science* 411(6), 854–864 (2010)
- [15] Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: *IEEE Congress on Evolutionary Computation, CEC 2003*, pp. 1918–1925 (2003)

- [16] Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. *Evolutionary Computation* 18(3), 335–356 (2010)
- [17] Glasmachers, T.: Optimized approximation sets of low-dimensional benchmark pareto fronts. In: 13th International Conference on Parallel Problem Solving from Nature, PPSN (2014)
- [18] Huband, S., Barone, L., While, R.L., Hingston, P.: A scalable multi-objective test problem toolkit. In: 3rd International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005, pp. 280–295 (2005)
- [19] Ponte, A., Paquete, L., Figueira, J.R.: On beam search for multicriteria combinatorial optimization problems. In: 9th International Conference in Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2012, pp. 307–321 (2012)
- [20] Tran, R., Wu, J., Denison, C., Ackling, T., Wagner, M., Neumann, F.: Fast and effective multi-objective optimisation of wind turbine placement. In: Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 1381–1388 (2013)
- [21] Vaz, D., Paquete, L., Ponte, A.: A note on the ε -indicator subset selection. *Theoretical Computer Science* 499, 113–116 (2013)
- [22] Wagner, M., Day, J., Neumann, F.: A fast and effective local search algorithm for optimizing the placement of wind turbines. *Renewable Energy* 51, 64–70 (2013)
- [23] Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
- [24] Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation* 3, 257–271 (1999)
- [25] Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* 8(2), 173–195 (2000)
- [26] Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, EUROGEN 2001, pp. 95–100 (2002)

A Provably Asymptotically Fast Version of the Generalized Jensen Algorithm for Non-dominated Sorting

Maxim Buzdalov and Anatoly Shalyto

ITMO University
49 Kronverkskiy prosp.
197101 Saint-Petersburg, Russia
buzdalov@rain.ifmo.ru, shalyto@mail.ifmo.ru

Abstract. The non-dominated sorting algorithm by Jensen, generalized by Fortin et al to handle the cases of equal objective values, has the running time complexity of $O(N \log^{K-1} N)$ in the general case. Here N is the number of points, K is the number of objectives and K is thought to be a constant when N varies. However, the complexity was not proven to be the same in the worst case.

A slightly modified version of the algorithm is presented, for which it is proven that its worst-case running time complexity is $O(N \log^{K-1} N)$.

Keywords: Non-dominated sorting, worst-case running time complexity, multi-objective optimization.

1 Introduction

In the K -dimensional space, a point $A = (a_1, \dots, a_K)$ is said to *dominate* a point $B = (b_1, \dots, b_K)$ when for all $1 \leq i \leq K$ it holds that $a_i \leq b_i$ and there exists j such that $a_j < b_j$. *Non-dominated sorting* of points in the K -dimensional space is a procedure of marking all points which are not dominated by any other point with the *rank* of 0, all points which are dominated by at least one point of the rank of 0 are marked with the rank of 1, all points which are dominated by at least one point of the rank $i - 1$ are marked with the rank of i .

Many well-known and widely used multi-objective evolutionary algorithms use the procedure of non-dominated sorting, or the procedure of determining the non-dominated solutions, which can be reduced to non-dominated sorting. These algorithms include NSGA-II [6], PESA [5], PESA-II [4], SPEA2 [11], PAES [9], PDE [2], and many more. The time complexity of a single iteration of these algorithms is often dominated by the complexity of a non-dominated sorting algorithm, so optimization of the latter makes such multi-objective evolutionary algorithms faster.

In Kung et al [10], the algorithm for determining the non-dominated solutions is proposed with the complexity of $O(N \log^{K-1} N)$, where N is the number of points and K is the dimension of the space. It is possible to use this algorithm to

perform non-dominated sorting: first, the non-dominated solutions are found and assigned the rank of 0. Then, these solutions are removed, the non-dominated solutions from the remaining ones are found and assigned the rank of 1. The process repeats until all the solutions are removed. This leads to the complexity of $O(N^2 \log^{K-1} N)$ in the worst case, if the maximum rank of a point in the result is $O(N)$.

Jensen [8] was the first to propose an algorithm for non-dominated sorting with the complexity of $O(N \log^{K-1} N)$. However, his algorithm was developed for the assumption that no two points share a common value for any objective, and the complexity was proven for the same assumption. The first attempt to fix this issue belongs, to the best of the authors' knowledge, to Fortin et al [7]. The corrected (or, as in [7], "generalized") algorithm works in all cases, and for the general case the performance is still $O(N \log^{K-1} N)$, but the only upper bound that was proven for the worst case is $O(N^2 K)$.

We present a modified version of this algorithm, for which we prove that its running time is $O(N \log^{K-1} N)$ for fixed K . The rest of the paper is structured as follows. Section 2 contains the description of the new algorithm. In Section 3, the proof of the worst-case running time complexity is given. In Section 4, some words are given on the performance of the original algorithm by Fortin et al. Section 5 concludes.

2 Algorithm Description

In this section, the modified algorithm is described. We try to be as much compatible with the notation of the original paper [7] as possible.

Multi-objective evolutionary algorithms work with candidate solutions to the problem they solve, or *individuals*. Each individual is evaluated and assigned a *fitness*. The fitness $q = f(p)$ of the individual p in a multi-objective problem is a vector of K objectives. We assume that the objectives are to be minimized.

The individuals along with their fitnesses are the input data for the non-dominated sorting algorithm. The algorithm should assign a rank to each individual, as described in Section 1. As two individuals with the same fitness will be assigned the same rank, the algorithm first groups the individuals by their fitnesses, and then works directly with fitnesses, assuming that no two fitnesses are equal in all objectives.

Let q_k be the k -th objective of the fitness q , $q_{1:k}$ be the first k objectives. The relation $q_{1:k} \prec t_{1:k}$ means that q dominates over t in first k objectives. Throughout all the listings in this paper, we denote as $\text{RANK}(q)$ the rank assigned to the fitness q .

The original algorithm from Fortin et al [7] makes use of the following procedures:

- $\text{NONDOMINATEDSORT}(P, K)$, the main procedure, receives a population P where each individual has a fitness with K objectives and returns the result of the non-dominated sorting — a sequence of Pareto fronts on which the given individuals reside. It does some preparation work, invokes NDHELPERA and constructs the answer.

```

1: procedure NDHELPERA( $S, k$ )
2:   if  $|S| < 2$  then return
3:   else if  $|S| = 2$  then
4:      $\{s^{(1)}, s^{(2)}\} \leftarrow S$ 
5:     if  $s_{1:k}^{(1)} < s_{1:k}^{(2)}$  then
6:        $\text{RANK}(s^{(2)}) \leftarrow \max\{\text{RANK}(s^{(2)}), \text{RANK}(s^{(1)}) + 1\}$ 
7:     end if
8:   else if  $k = 2$  then
9:     SWEEPA( $S$ )
10:  else if  $|\{s_k | s \in S\}| = 1$  then
11:    NDHELPERA( $S, k - 1$ )
12:  else
13:     $L, M, H \leftarrow \text{SPLITBY}(S, \text{median}\{s_k | s \in S\}, k)$ 
14:    NDHELPERA( $L, k$ )
15:    NDHELPERB( $L, M, k - 1$ )
16:    NDHELPERA( $M, k - 1$ )
17:    NDHELPERB( $L \cup M, H, k - 1$ )
18:    NDHELPERA( $H, k$ )
19:  end if
20: end procedure

```

Fig. 1. The procedure NDHELPERA. It assigns ranks to fitnesses from S using the first k objectives.

```

1: procedure NDHELPERB( $L, H, k$ )
2:   if  $L = \{\}$  or  $H = \{\}$  then return
3:   else if  $|L| = 1$  or  $|H| = 1$  then
4:     for all  $h \in H, l \in L$  do
5:       if  $l_{1:k} \leq h_{1:k}$  then
6:          $\text{RANK}(h) \leftarrow \max\{\text{RANK}(h), \text{RANK}(l) + 1\}$ 
7:       end if
8:     end for
9:   else if  $k = 2$  then
10:    SWEEPB( $L, H$ )
11:   else if  $\max\{l_k | l \in L\} \leq \min\{h_k | h \in H\}$  then
12:    NDHELPERB( $L, H, k - 1$ )
13:   else if  $\min\{l_k | l \in L\} \leq \max\{h_k | h \in H\}$  then
14:     $m \leftarrow \text{median}\{s_k | s \in L \cup H\}$ 
15:     $L_1, M_1, H_1 \leftarrow \text{SPLITBY}(L, m, k)$ 
16:     $L_2, M_2, H_2 \leftarrow \text{SPLITBY}(H, m, k)$ 
17:    NDHELPERB( $L_1, L_2, k$ )
18:    NDHELPERB( $L_1, M_2, k - 1$ )
19:    NDHELPERB( $M_1, M_2, k - 1$ )
20:    NDHELPERB( $L_1 \cup M_1, M_2, k - 1$ )
21:    NDHELPERB( $M_1, M_2, k$ )
22:   end if
23: end procedure

```

Fig. 2. The procedure NDHELPERB. It adjusts ranks of fitnesses from H using the first k objectives by comparing them with fitnesses from L .

```

1: procedure SPLITBY( $S, m, k$ )
2:    $L \leftarrow \{s \in S \mid s_k < m\}$ 
3:    $M \leftarrow \{s \in S \mid s_k = m\}$ 
4:    $H \leftarrow \{s \in S \mid s_k > m\}$ 
5:   return  $L, M, H$ 
6: end procedure

```

Fig. 3. The generic split procedure. In each resulting set, the original order of elements is preserved.

- NDHELPERA(S, k) assigns ranks to the fitnesses from the set S based on the first k objectives. It is recursive and may call itself, NDHELPERB, SWEEPA and SPLITA.
- NDHELPERB(L, H, k) assigns ranks to the fitnesses from the set H by comparing them to the fitnesses from the set L based on the first k objectives. It is recursive and may call itself, SWEEPB and SPLITB.
- SWEEPA(S) assigns ranks to the fitnesses from the set S based on the first two objectives. It is implemented using the sweep-line approach, and its running time complexity is $O(|S| \log |S|)$.
- SWEEPB(L, H) assigns ranks to the fitnesses from the set H by comparing them to the fitnesses from the set L based on the first two objectives. It is implemented using the sweep-line approach, and its running time complexity is $O((|L| + |H|) \log |L|)$.
- SPLITA(S, k) partitions the set of fitnesses S in two sets around its median for the objective k and balances the resulting sets using the elements equal to the median.
- SPLITB(L, H, k) partitions the sets of fitnesses L and H into two sets each around the median of the largest set for the objective k and balances the resulting sets using the elements equal to the median.

Procedures NONDOMINATEDSORT, SWEEPA and SWEEPB remain the same as in Fortin et al [7]. The procedures SPLITA and SPLITB are not used by the modified algorithm.

We redefine the procedures NDHELPERA, which is shown in Fig. 1, and NDHELPERB, which is shown in Fig. 2. These procedures use the procedure SPLITBY, which is shown in Fig. 3.

The rest of this section concentrates on differences between the original algorithm by Fortin et al and the proposed algorithm. The correctness of the proposed algorithm can be proven in the same way as the correctness of the original one.

2.1 Splitting into Three Parts, NDHelperA

The difference between the modified NDHELPERA and its original version is that the set S is split into three parts (line 12–13) instead of two parts as in Fortin et al [7]. Generally, we have three sets:

- the set L of the elements with the k -th objective less than the median;
- the set M of the elements with the k -th objective equal to the median;
- the set H of the elements with the k -th objective greater than the median.

In the original algorithm, the set M was added either to L or to H , depending on their size. The correctness of the algorithm does not depend on the exact way of splitting. Consider that we split $S = (L \cup M) \cup H$, so that we call:

- NDHELPERA($L \cup M, k$);
- NDHELPERB($L \cup M, H, k - 1$);
- NDHELPERA(H, k).

It cannot be predicted without extra assumptions how the set $L \cup M$ will be split in the first recursive call to NDHELPERA. However, any split such that all values of the k -th objective in the left are strictly less than all such values on the right is valid, and if the algorithm used it, the correctness would be preserved. In particular, the set $L \cup M$ can be split into L and M . After that, NDHELPERA($L \cup M, k$) can be further expanded as follows:

- NDHELPERA(L, k);
- NDHELPERB($L, M, k - 1$);
- NDHELPERA(M, k).

However, it is known that the k -th objective in M is the same for all fitnesses, which means that NDHELPERA(M, k) can be rewritten as NDHELPERA($M, k - 1$). So finally the procedures are called in the following way:

- NDHELPERA(L, k);
- NDHELPERB($L, M, k - 1$);
- NDHELPERA($M, k - 1$);
- NDHELPERB($L \cup M, H, k - 1$);
- NDHELPERA(H, k).

This is exactly what one can see in lines 14–18 of the updated version of NDHELPERA.

2.2 Splitting into Three Parts, NDHelperB

The ideas from Section 2.1 can be applied to the procedure NDHELPERB as well. Assuming we have some pivot (in the original algorithm it is the median of the largest of the L and H sets), we have six sets after the split:

- L_1 — the elements of L with k -th objective less than the pivot;
- M_1 — the elements of L with k -th objective equal to the pivot;
- H_1 — the elements of L with k -th objective greater than the pivot;
- L_2 — the elements of H with k -th objective less than the pivot;
- M_2 — the elements of H with k -th objective equal to the pivot;
- H_2 — the elements of H with k -th objective greater than the pivot.

In the original algorithm, the sets M_1 and M_2 are joined either to L_1 and L_2 or to H_1 and H_2 , depending on the sizes of L_1, L_2, H_1, H_2 . Note that, although performance depends on the particular choice, correctness does not. To perform the analysis, we merge M_i with L_i . This results in the following sequence of recursive calls:

- NDHELPERB($L_1 \cup M_1, L_2 \cup M_2, k$);
- NDHELPERB($L_1 \cup M_1, H_2, k - 1$);
- NDHELPERB(H_1, H_2, k).

Using the same reasoning as in Section 2.1, the first call can be expanded to:

- NDHELPERB(L_1, L_2, k);
- NDHELPERB($L_1, M_2, k - 1$);
- NDHELPERB(M_1, M_2, k).

The last call can be rewritten as NDHELPERB($M_1, M_2, k - 1$) because all values of the k -th objective are the same in all fitnesses. So the final call sequence is:

- NDHELPERB(L_1, L_2, k);
- NDHELPERB($L_1, M_2, k - 1$);
- NDHELPERB($M_1, M_2, k - 1$);
- NDHELPERB($L_1 \cup M_1, H_2, k - 1$);
- NDHELPERB(H_1, H_2, k).

One can see exactly this sequence in lines 17–21 of the procedure NDHELPERB.

2.3 The Choice of Pivot

Consider the last case of NDHELPERB where the pivot is chosen. The original choice of Jensen [8], followed by Fortin et al [7], was to choose the pivot in such a way that the sizes of sets in the recursive calls did not exceed $3N/4$, where $N = |H| + |L|$. In the case of the proposed algorithm, it is more important to ensure the sizes of the recursive calls *with the same k* to be small, while not caring for the sizes of the middle sets M_1 and M_2 . This forced us to choose the pivot to be the median of k -th objectives of $L \cup H$, which ensures that $|L_1| + |L_2| \leq N/2$ and $|M_1| + |M_2| \leq N/2$.

3 Running Time Estimation

Let us outline the statements about the procedures above:

- In the last case of NDHELPERA, $|L| \leq N/2$ and $|H| \leq N/2$, where $N = |S|$. This is due to the choice of the pivot equal to the median of k -th objectives of elements from S .
- In the last case of NDHELPERB, $|L_1| + |L_2| \leq N/2$ and $|H_1| + |H_2| \leq N/2$, where $N = |L| + |H|$. As already noted in Section 2.3, this is due to the choice of the pivot.

This helps us to prove the necessary theorems.

3.1 Running Time of NDHelperB

Theorem 1. *The running time of NDHELPERB(L, H, k) is*

$$T_B(L, H, k) = O(N \log^{k-1} N)$$

for constant $k \geq 2$, where $N = |H| + |L|$.

Proof. Consider the boundary cases where $|L| \leq 1$ or $|H| \leq 1$. In this case, the running time is $\Theta(1)$ or $\Theta(Nk)$, both are $O(N \log N)$.

The non-boundary cases are proven using induction by k . The base case is $k = 2$. NDHELPERB($L, H, 2$) just calls SWEEPB(L, H), which is $O((|L| + |H|) \log |L|) = O(N \log N)$.

For $k > 2$, assume that the induction statement is proven for $k - 1$. In the worst case, the running time of NDHELPER(L, H, k) is:

$$\begin{aligned} T_B(L, H, k) &= T_B(L_1, L_2, k) + \\ &\quad + T_B(L_1, M_2, k - 1) + \\ &\quad + T_B(M_1, M_2, k - 1) + \\ &\quad + T_B(L_1 \cup M_1, H_2, k - 1) + \\ &\quad + T_B(H_1, H_2, k) + \\ &\quad + \Theta(|L| + |H|). \end{aligned}$$

where the $\Theta(|L| + |H|)$ addend corresponds to finding the pivot and splitting the sets. The addends 2–4 and 6 in the expression above can be summarized as $O(N \log^{k-2} N)$ by the induction assumption. It is known that $|L_1| + |L_2| \leq N/2$ and $|H_1| + |H_2| \leq N/2$. If we rewrite $T_B(A, B, k)$ as $T'_B(|A| + |B|, k)$, then:

$$T_B(L, H, k) = T'_B(N, k) \leq 2T'_B(N/2, k) + O(N \log^{k-2} N).$$

From the results of Bentley [3], it follows that:

$$T_B(L, H, k) = O(N \log^{k-1} N). \square$$

3.2 Running Time of NDHelperA

Theorem 2. *The running time of NDHELPERA(S, k) is*

$$T_A(S, k) = O(N \log^{k-1} N)$$

for constant $k \geq 2$, where $N = |S|$.

Proof. In the boundary case of $|S| \leq 2$, the running time is $\Theta(k) = \Theta(1)$.

The non-boundary cases are proven using induction by k . The base case is $k = 2$. NDHELPERA($S, 2$) calls SWEEPA(S), which is $O(|S| \log |S|) = O(N \log N)$.

For $k > 2$, assume that the induction statement is proven for $k - 1$. The running time of NDHELPERA(S, k) is at most:

$$\begin{aligned}
 T_A(S, k) &= T_A(L, k) + \\
 &\quad + T_B(L, M, k - 1) + \\
 &\quad + T_A(M, k - 1) + \\
 &\quad + T_B(L \cup M, H, k - 1) + \\
 &\quad + T_A(H, k) + \\
 &\quad + \Theta(|S|).
 \end{aligned}$$

where the $\Theta(|S|)$ addend corresponds to finding the pivot and splitting the sets. The addends 2-4 and 6 can be summarized as $O(N \log^{k-2} N)$ by the induction assumption. It is known that $|L| \leq N/2$ and $|H| \leq N/2$. Using [3], we find that:

$$\begin{aligned}
 T_A(S, k) &= T'_A(N, k) \leq 2T'_A(N/2, k) + O(N \log^{k-2} N) = \\
 &= O(N \log^{k-1} N). \square
 \end{aligned}$$

From the last theorem it immediately follows that the running time of the whole algorithm is $O(N \log^{K-1} N)$.

4 Discussion

In this section, two topics are discussed. First, the worst case given in [7] is shown to be not only “very unlikely”, but not happening when K is thought to be a constant. Second, some ideas are given that may help to apply the proof above, with some more detailed case analysis, to the original algorithm.

4.1 The Worst-Case Bound for the Original Algorithm

In [7], the worst-case running time estimation for the original algorithm is deduced from the following recurrence:

$$\begin{aligned}
 T_A(N, K) &= T_A(N - 1, K) + T_A(1, K) + \\
 &\quad + T_B(N - 1, 1, K - 1) + \Theta(N).
 \end{aligned}$$

Such split can only occur in the case when there is only one element larger than the median. As the median elements are always merged to the set which size is smaller, there is at most one element smaller than the median. We can predict that $T_A(N - 1, K)$ will either be split as “one smaller element” and “ $N - 2$ median elements”, or just be delegated to $T_A(N - 1, K - 1)$.

The actual situation is that, in such heavily imbalanced cases, K decreases as N decreases. This makes the “worst cases” be implementable only if $K = \Theta(N)$. As an example, one may construct the following test case:

- (0, 0, 0, ..., 0, 0, 1),
- (0, 0, 0, ..., 0, 1, 1),
- ...
- (0, 0, 1, ..., 1, 1, 1),
- (0, 1, 1, ..., 1, 1, 1),
- (1, 1, 1, ..., 1, 1, 1).

Here the recurrence for the running time for both the original algorithm and the modified version appears to be:

$$T_A(N, K) = T_A(N - 1, K - 1) + T_A(1, K) + T_B(N - 1, 1, K - 1) + \Theta(N).$$

The solution can be written as $T_A(N, K) = \Theta(N^2K)$. However, $N = K$, so K is not a constant anymore. In the situation of K depending on N , the analysis which produced the expression $O(N \log^{K-1} N)$ actually gives another result, so comparison of the expressions for constant K and for varying K is not valid.

As a final remark to this section, the “proof” of the worst-case performance in the original paper does not prove that there exists a sequence of tests with constant K and growing N , such that the running time of the algorithm grows as $O(N^2K)$. It just considers a class of tests where $K = \Theta(N)$.

4.2 Applicability of the Proof to the Original Algorithm

The authors believe that the proof similar to the one given in this paper can be constructed for the original algorithm from [7]. The reason is that, in the heavily imbalanced cases (when $|M| > \max(|L|, |H|)$ for NDHELPERA, and when $|M_1| > \max(|L_1|, |H_1|)$ and $|M_2| > \max(|L_2|, |H_2|)$ for NDHELPERB, see Fig. 1 and Fig. 2) the splits in both procedures go in exactly the same way as in the proposed algorithm. As the running time complexity in both well balanced and heavily imbalanced cases is the same, the authors think that the overall complexity is the same.

The immediate application of the proof to the original algorithm cannot be performed because the splits in a partially balanced case for NDHELPERB are quite unpredictable due to the algorithm for pivot selection. The proper analysis of such case is welcome.

5 Conclusion

A new version of the non-dominated sorting algorithm is presented. The ideas of the algorithm were first presented by Jensen [8] and corrected by Fortin et al [7] to handle the cases of equal objectives. The presented version differs from the latter in the way the splits in the procedures NDHELPERA and NDHELPERB are performed, as well as in the way the pivot is selected in NDHELPERB.

We proved the running time complexity is $O(N \log^{K-1} N)$ in the worst case for this version.

The implementation of the algorithm, along with some benchmarks, is available at GitHub [1].

This work was financially supported by the Government of Russian Federation, Grant 074-U01.

References

1. Source code for the implementation (a part of this paper), <https://github.com/mbuzdalov/papers/tree/master/2014-ppsn-jensen-fortin>
2. Abbass, H.A., Sarker, R., Newton, C.: PDE: A Pareto Frontier Differential Evolution Approach for Multiobjective Optimization Problems. In: Proceedings of the Congress on Evolutionary Computation, pp. 971–978. IEEE Press (2001)
3. Bentley, J.L.: Multidimensional Divide-and-conquer. *Communications of ACM* 23(4), 214–229 (1980)
4. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 283–290. Morgan Kaufmann Publishers (2001)
5. Corne, D.W., Knowles, J.D., Oates, M.J.: The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN VI. LNCS, vol. 1917, pp. 839–848. Springer, Heidelberg (2000)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *Transactions on Evolutionary Computation* 6, 182–197 (2000)
7. Fortin, F.A., Grenier, S., Parizeau, M.: Generalizing the Improved Run-time Complexity Algorithm for Non-dominated Sorting. In: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 615–622. ACM (2013)
8. Jensen, M.T.: Reducing the Run-time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. *Transactions on Evolutionary Computation* 7(5), 503–515 (2003)
9. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation* 8(2), 149–172 (2000)
10. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. *Journal of ACM* 22(4), 469–476 (1975)
11. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Proceedings of the EUROGEN 2001 Conference, pp. 95–100 (2001)

Clustering-Based Selection for Evolutionary Many-Objective Optimization

Roman Denysiuk^{1,3}, Lino Costa^{2,3}, and Isabel Espírito Santo^{2,3}

¹ Algoritmi R&D Center

roman.denysiuk@algoritmi.uminho.pt

² Department of Production and Systems Engineering

³ University of Minho, Braga, Portugal

{lac,iapinho}@dps.uminho.pt

Abstract. This paper discusses a selection scheme allowing to employ a clustering technique to guide the search in evolutionary many-objective optimization. The underlying idea to avoid the curse of dimensionality is based on transforming the objective vectors before applying a clustering and the selection of cluster representatives according to the distance to a reference point. The experimental results reveal that the proposed approach is able to effectively guide the search in high-dimensional objective spaces, producing highly competitive performance when compared with state-of-the-art algorithms.

1 Introduction

As problems with a large number of objectives become widespread in practice, the issue of dealing with many-objective problems has gained a significant attention in the evolutionary multiobjective optimization (EMO) community. Some researchers suggest to handle many-objective problems by modifying the Pareto dominance relation, assigning different ranks to nondominated solutions, using the decision maker's preferences during the search, incorporating the quality indicators or scalarizing functions into the fitness assignment, or reducing the problem's dimensionality whenever redundant objectives are identified. A good review of such approaches can be found in [1]. Despite the recent advances in solving many-objective problems, there are a number of disadvantages related to such approaches. In particular, the practical application of the hypervolume, which has nice mathematical properties, is limited due to the high computational cost. The use of a scalarizing fitness assignment necessitates a set of weight vectors to be provided in advance, being not always an easy task especially for high dimensions. The use of preference information during the search allows to find only certain regions of the Pareto front, whereas dimensionality reduction techniques are only suitable for problems having redundant objectives. Thus, the need for efficient and self-adaptive methodologies persists.

In this work, we focus on improving the scalability of Pareto-dominance based algorithms and argue that a clustering-based diversity maintenance can be successfully used for solving many-objective problems. For this purpose, we propose to perform a transformation on objective vectors, aimed at reducing the

distances between solutions in high-dimensional spaces. So that a clustering can group those solutions, which are distant in the original space but can be viewed as representatives of similar regions of the fitness landscape. The necessary selection pressure is provided by minimizing the distances of population members to a reference point.

The remainder of this paper is organized as follows. Section 2 describes an evolutionary algorithm with the proposed selection scheme. Section 3 presents the results of a comparison study and discusses variants of the suggested selection. Section 4 concludes the work and outlines further research opportunities.

2 Algorithm

In the following, we present an evolutionary many-objective optimization algorithm with clustering-based selection (EMyO/C), considering an optimization problem of the form:

$$\underset{\mathbf{x} \in \Omega \subset \mathbb{R}^n}{\text{minimize:}} \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))^T, \tag{1}$$

where m - is the number objective and n - is the number of variables.

EMyO/C reflects a general framework of an evolutionary algorithm with the $(\mu + \lambda)$ selection scheme, where the mating selection, variation and environmental selection are successively applied during the generation. In EMyO/C, an initial population is randomly generated and a reference point, \mathbf{z} , initialized as: $\forall j \in \{1, \dots, m\} : z_j = \min_{1 \leq i \leq \mu} f_j(\mathbf{x}^i)$. In the mating selection, each population member is selected to the mating pool. The variation procedure relies on the DE operator, adopting the idea presented in [2]. For each individual in the mating pool, two different individuals are randomly selected. A difference vector, \mathbf{v} , is calculated using these individuals. To introduce additional variation, the polynomial mutation [3] is applied on the difference vector, \mathbf{v} . The resulting difference vector is restricted as follows:

$$v_j = \begin{cases} -\delta_j & \text{if } v_j < -\delta_j \\ \delta_j & \text{if } v_j > \delta_j \\ v_j & \text{otherwise} \end{cases} \tag{2}$$

where

$$\delta_j = \frac{ub_j - lb_j}{2} \quad \forall j \in \{1, \dots, n\}, \tag{3}$$

ub_j and lb_j are the upper and lower bounds of the j -th variable, respectively. An offspring, \mathbf{x}' , is generated by mutating the parent individual, \mathbf{x} , as:

$$x'_j = \begin{cases} x_j + v_j & \text{if } rand < CR \\ x_j & \text{otherwise} \end{cases} \quad \forall j \in \{1, \dots, n\}. \tag{4}$$

To ensure the offspring feasibility, it is repaired as:

$$x'_j = \min\{\max\{x'_j, lb_j\}, ub_j\} \quad \forall j \in \{1, \dots, n\}. \tag{5}$$

The resulting offspring, \mathbf{x}' , is compared with its parent, \mathbf{x} . If there is a difference in at least one gene, then the offspring is evaluated and added to the offspring population. Otherwise, two other individuals are selected from the mating pool, and the above described steps - including computation of \mathbf{v} , mutation restriction, and creation of \mathbf{x}' - are performed until an individual different from \mathbf{x} is produced. This allows to avoid a situation in which the offspring identical to its parent is evaluated and added to the population. It should be noted that applying the polynomial mutation allows to produce new genotypes even when the whole population is converged to a single solution. Further, each time an offspring is evaluated, the components of the reference point are updated if there are smaller objective values. The environmental selection performs the nondominated sorting procedure [3] and selects a new population from the multiset of parents and offspring, according to the rank values. In a case where the last accepted front, \mathcal{F}_l , cannot be completely accommodated, the truncation procedure is performed to select k best individuals as follows.

1. For each individual in \mathcal{F}_l , the Euclidean distance in the objective space to the reference point, \mathbf{z} , is calculated.
2. For each individual in \mathcal{F}_l , the objectives are translated as:

$$f_i = f_i - z_i \quad \forall i \in \{1, \dots, m\}. \quad (6)$$

3. Each individual in \mathcal{F}_l is projected onto the unit hyperplane as:

$$f_i = f_i / \sum_{j=1}^m f_j \quad \forall i \in \{1, \dots, m\}. \quad (7)$$

4. Using projected individuals, k clusters are formed as follows:

Step 1 Initially, each individual is treated as a separate cluster

$$C = \{C_1, C_2, \dots, C_{|\mathcal{F}_l|}\}.$$

Step 2 If $|C| = k$, stop. Otherwise, go to Step 3.

Step 3 For each pair of clusters, the distance between two clusters, d_{12} , is calculated as:

$$d_{12} = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} d(i, j),$$

where $d(i, j)$ is the Euclidean distance between individuals i and j .

Step 4 The pair of clusters having the smallest distance is merged.

Go to Step 2.

5. In each cluster, a representative is selected and added to the new population, where a cluster representative is an individual having the smallest distance to the reference point.

It should be noted that the proposed selection procedure does not require any additional parameter, being completely adaptive and easy to implement. The complexity of the clustering algorithm is mainly governed by the number of points in \mathcal{F}_l , whereas it is polynomial in the number of objectives. This feature is especially attractive to solve problems with a large number of objectives.

3 Performance Assessment

To validate EMyO/C, it is compared with IBEA [4], MOEA/D [5], MSOPS [6], MSOPS2 [7], and HypE [8] on the DTLZ test suite [9] with 30 decision variables having between 2 and 20 objectives.

3.1 Performance Indicators and Statistical Comparison

The outcomes produced by the algorithms are assessed using the unary additive epsilon ($\epsilon+$) indicator [10], the hypervolume (HV) indicator [11], and the inverted generational distance (IGD) indicator [12]. To calculate the $\epsilon+$ and IGD indicators, for all problems, 1,000 uniformly distributed points along the Pareto front are generated. To calculate the HV indicator, the nadir point is used as a reference point. Solutions that do not dominate the nadir point are discarded. If there is no solution dominating the nadir point, then the hypervolume of the approximation set is equal to zero. Further, solutions used to calculate the hypervolume are normalized using the ideal and nadir points. When the number of objectives is more than 6 the hypervolume is approximated using 10^6 uniformly sampled points, otherwise the hypervolume is computed exactly.

To provide the results with statistical confidence, the single-problem analysis is performed using the Wilcoxon rank-sum test. The multiple-problem analysis is performed on algorithms' ranks using the Friedman test to determine whether there is a significant difference among the results, and the Bonferroni procedure for a post-hoc statistical analysis to detect concrete differences among the algorithms [13]. All tests are performed at significance level of $\alpha = 0.05$.

3.2 Experimental Setup

EMyO/C is implemented and tested in JavaTM, whereas IBEA and HypE are used within the PISA [14] framework¹, MOEA/D is used within the jMetal [15] framework², and the implementations of MSOPS and MSOPS2 are taken from the author's web page³.

For each algorithm, 30 independent runs are performed on each problem with a population size of $\mu = 300$, running for 500 generations. The other parameter settings for EMyO/C are: $CR = 0.15$, $\eta_m = 20$, and $p_m = 1/n$ (n is the number of decision variables). The other algorithms use the default settings, except common parameters with EMyO/C, which use the same values to guarantee a fair comparison.

3.3 Performance Comparison

Table 1 presents the median values of the quality indicators and the statistical comparison of the algorithms. In the table, the small values of the $\epsilon+$ indicator

¹ available at <http://www.tik.ee.ethz.ch/pisa>

² available at <http://jmetal.sourceforge.net>

³ available at <http://code.evanhughes.org>

Table 1. Median values of the epsilon (the lower the better), hypervolume (the higher the better), and IGD (the lower the better) indicators after 30 runs. The superscripts 1, 2, 3, 4, 5, and 6 indicate whether the respective algorithm performs significantly better than EMyO/C, IBEA, MOEA/D, MSOPS, MSOPS2, and HypE, respectively.

	EMyO/C	IBEA	MOEA/D	MSOPS	MSOPS2	HypE	
2-objectives							
DTLZ1	$\epsilon+$	0.002 ^{2,3,4,5,6}	0.21 ^{3,4,5}	7.696 ⁵	6.843 ⁵	14.489	0.034 ^{2,3,4,5}
	HV	0.498 ^{2,3,4,5,6}	0.136 ^{3,4,5}	0	0	0	0.439 ^{2,3,4,5}
	IGD	0.001 ^{2,3,4,5,6}	0.121 ^{3,4,5}	10.024 ⁵	9.099 ⁵	19.336	0.02 ^{2,3,4,5}
DTLZ2	$\epsilon+$	0.005 ^{2,4,6}	0.006 ^{4,6}	0.002 ^{1,2,4,5,6}	0.007 ⁶	0.004 ^{1,2,4,6}	0.046
	HV	0.213 ^{4,6}	0.213 ^{4,6}	0.213 ^{1,2,4,6}	0.212 ⁶	0.213 ^{1,2,3,4,6}	0.203
	IGD	0.002 ^{2,4,6}	0.005 ⁶	0.001 ^{1,2,4,5,6}	0.002 ^{2,6}	0.001 ^{1,2,4,6}	0.009
DTLZ3	$\epsilon+$	0.005 ^{2,3,4,5,6}	0.341 ^{3,4,5}	22.478 ⁵	28.001 ⁵	40.545	0.12 ^{2,3,4,5}
	HV	0.213 ^{2,3,4,5,6}	0	0	0	0	0.13 ^{2,3,4,5}
	IGD	0.002 ^{2,3,4,5,6}	0.397 ^{3,4,5}	27.579 ^{4,5}	37.67 ⁵	48.984	0.061 ^{2,3,4,5}
DTLZ4	$\epsilon+$	0.005 ^{2,4,5,6}	0.005 ^{4,5,6}	0.002 ^{1,2,4,5,6}	0.009 ^{5,6}	0.265	0.026 ⁵
	HV	0.213 ^{4,5,6}	0.213 ^{1,4,5,6}	0.213 ^{1,2,4,5,6}	0.212 ^{5,6}	0.051	0.208 ⁵
	IGD	0.002 ^{2,4,5,6}	0.005 ⁵	0.001 ^{1,2,4,5,6}	0.002 ^{2,5,6}	0.226	0.005 ⁵
DTLZ7	$\epsilon+$	0.006 ^{2,4,5,6}	0.018 ^{5,6}	0.004 ^{1,2,4,5,6}	0.011 ^{2,5,6}	3.77	0.093 ⁵
	HV	0.336 ^{2,3,4,5,6}	0.335 ^{4,5,6}	0.336 ^{2,4,5,6}	0.33 ^{5,6}	0	0.293 ⁵
	IGD	0.002 ^{2,3,4,5,6}	0.007 ^{4,5,6}	0.002 ^{2,4,5,6}	0.008 ^{5,6}	2.945	0.136 ⁵
3-objectives							
DTLZ1	$\epsilon+$	0.021 ^{2,3,4,5,6}	0.276 ^{3,4,5,6}	9.399 ⁵	4.025 ^{3,5}	100.495	1.187 ^{3,4,5}
	HV	0.806 ^{2,3,4,5,6}	0.265 ^{3,4,5,6}	0	0	0	0
	IGD	0.012 ^{2,3,4,5,6}	0.187 ^{3,4,5,6}	14.458 ⁵	5.884 ^{3,5}	133.662	1.333 ^{3,4,5}
DTLZ2	$\epsilon+$	0.051 ^{3,4,6}	0.047 ^{1,3,4,6}	0.087 ⁶	0.066 ^{3,6}	0.043 ^{1,2,3,4,6}	0.121
	HV	0.439 ^{3,4,6}	0.441 ^{1,3,4,6}	0.419 ⁶	0.437 ^{3,6}	0.442 ^{1,2,3,4,6}	0.397
	IGD	0.032 ^{2,3,4,6}	0.061 ⁶	0.039 ^{2,6}	0.034 ^{2,3,6}	0.031 ^{1,2,3,4,6}	0.077
DTLZ3	$\epsilon+$	0.053 ^{2,3,4,5,6}	0.47 ^{3,4,5,6}	20.974 ⁵	17.645 ⁵	244.304	6.717 ^{3,4,5}
	HV	0.439 ^{2,3,4,5,6}	0	0	0	0	0
	IGD	0.033 ^{2,3,4,5,6}	0.553 ^{3,4,5,6}	28.443 ⁵	27.796 ⁵	327.555	7.002 ^{3,4,5}
DTLZ4	$\epsilon+$	0.053 ^{3,4,5,6}	0.047 ^{1,3,4,5,6}	0.081 ^{5,6}	0.06 ^{3,5,6}	0.634	0.098 ⁵
	HV	0.437 ^{3,4,5,6}	0.441 ^{1,3,4,5,6}	0.425 ^{5,6}	0.436 ^{3,5,6}	0.241	0.414 ⁵
	IGD	0.032 ^{2,3,4,5,6}	0.06 ^{5,6}	0.039 ^{2,5,6}	0.034 ^{2,3,5,6}	0.247	0.063 ⁵
DTLZ7	$\epsilon+$	0.051 ^{2,3,4,5,6}	0.057 ^{3,4,5,6}	0.167 ^{5,6}	0.132 ^{3,5,6}	5.044	0.428 ⁵
	HV	0.364 ^{2,3,4,5,6}	0.361 ^{3,4,5,6}	0.309 ^{5,6}	0.312 ^{3,5,6}	0	0.23 ⁵
	IGD	0.037 ^{2,3,4,5,6}	0.092 ^{3,4,5,6}	0.103 ^{4,5,6}	0.158 ^{5,6}	3.277	0.37 ⁵
5-objectives							
DTLZ1	$\epsilon+$	0.075 ^{2,3,4,5,6}	0.325 ^{3,4,5,6}	7.619 ^{5,6}	2.16 ^{3,5,6}	102.255	15.715 ⁵
	HV	0.93 ^{2,3,4,5,6}	0.409 ^{3,4,5,6}	0	0	0	0
	IGD	0.069 ^{2,3,4,5,6}	0.243 ^{3,4,5,6}	11.502 ^{5,6}	3.344 ^{3,5,6}	159.717	18.451 ⁵
DTLZ2	$\epsilon+$	0.144 ^{3,4,6}	0.131 ^{1,3,4,5,6}	0.18 ⁶	0.159 ^{3,6}	0.135 ^{1,3,4,6}	0.377
	HV	0.705 ^{3,4,5,6}	0.725 ^{1,3,4,5,6}	0.651 ⁶	0.7 ^{3,5,6}	0.696 ^{3,6}	0.343
	IGD	0.164 ^{2,3,4,6}	0.21 ^{4,6}	0.203 ^{2,4,6}	0.216 ⁶	0.159 ^{1,2,3,4,6}	0.408
DTLZ3	$\epsilon+$	0.143 ^{2,3,4,5,6}	0.584 ^{3,4,5,6}	16.688 ^{5,6}	11.724 ^{3,5,6}	325.935	32.866 ⁵
	HV	0.703 ^{2,3,4,5,6}	0	0	0	0	0
	IGD	0.164 ^{2,3,4,5,6}	0.71 ^{3,4,5,6}	20.895 ^{5,6}	16.946 ^{3,5,6}	464.898	32.464 ⁵
DTLZ4	$\epsilon+$	0.144 ^{3,4,5,6}	0.131 ^{1,3,4,5,6}	0.183 ⁶	0.158 ^{3,6}	0.174 ⁶	0.213
	HV	0.703 ^{3,5,6}	0.727 ^{1,3,4,5,6}	0.666 ⁶	0.702 ^{3,5,6}	0.662 ⁶	0.604
	IGD	0.168 ^{2,3,4,5,6}	0.21 ^{3,4,6}	0.246 ⁶	0.229 ^{3,6}	0.18 ^{2,3,4,6}	0.253
DTLZ7	$\epsilon+$	0.231 ^{4,5,6}	0.257 ^{4,5,6}	0.217 ^{4,5,6}	0.46 ^{5,6}	8.895	0.832 ⁵
	HV	0.323 ^{3,4,5,6}	0.337 ^{1,3,4,5,6}	0.271 ^{4,5,6}	0.199 ⁵	0	0.217 ^{4,5}
	IGD	0.261 ^{2,3,4,5,6}	0.385 ^{3,4,5,6}	0.413 ^{4,5,6}	0.518 ^{5,6}	4.913	0.648 ⁵
10-objectives							
DTLZ1	$\epsilon+$	0.212 ^{2,3,4,5,6}	0.334 ^{3,4,5,6}	2.547 ^{5,6}	1.151 ^{3,5,6}	65.446	11.561 ⁵
	HV	0.484 ^{3,4,5,6}	0.729 ^{1,3,4,5,6}	0	0	0	0
	IGD	0.272 ^{3,4,5,6}	0.304 ^{3,4,5,6}	3.478 ^{5,6}	1.766 ^{3,5,6}	108.78	17.875 ⁵
DTLZ2	$\epsilon+$	0.249 ^{2,3,4,5,6}	0.259 ^{3,5,6}	0.289 ^{5,6}	0.259 ^{3,5,6}	0.364 ⁶	0.656
	HV	0.917 ^{3,4,5,6}	0.923 ^{1,3,4,5,6}	0.867 ^{5,6}	0.911 ^{3,5,6}	0.821 ⁶	0.348
	IGD	0.442 ^{2,3,4,6}	0.481 ^{4,6}	0.454 ^{2,4,6}	0.49 ⁶	0.415 ^{1,2,3,4,6}	0.679
DTLZ3	$\epsilon+$	0.411 ^{2,3,4,5,6}	0.632 ^{3,4,5,6}	5.628 ^{5,6}	6.045 ^{5,6}	209.395	25.339 ⁵
	HV	0.55 ^{2,3,4,5,6}	0	0	0	0	0
	IGD	0.547 ^{2,3,4,5,6}	0.849 ^{3,4,5,6}	7.138 ^{4,5,6}	8.762 ^{5,6}	266.516	25.63 ⁵
DTLZ4	$\epsilon+$	0.267 ^{3,5,6}	0.248 ^{1,3,4,5,6}	0.322 ⁶	0.27 ^{3,5,6}	0.282 ^{3,6}	0.988
	HV	0.926 ^{3,4,5,6}	0.931 ^{1,3,4,5,6}	0.888 ⁶	0.916 ^{3,5,6}	0.891 ⁶	0
	IGD	0.495 ^{3,4,5,6}	0.475 ^{1,3,4,5,6}	0.565 ⁶	0.545 ^{3,6}	0.521 ^{3,4,6}	1.087
DTLZ7	$\epsilon+$	0.727 ^{2,3,4,5,6}	0.791 ^{4,5,6}	0.762 ^{4,5,6}	0.826 ^{5,6}	13.077	0.832 ⁵
	HV	0.019 ⁵	0.183 ^{1,3,4,5,6}	0.022 ⁵	0.033 ^{1,3,5}	0	0.112 ^{1,3,4,5}
	IGD	0.947 ^{3,4,5,6}	0.97 ^{3,4,5,6}	1.021 ^{4,5,6}	1.622 ⁵	7.274	1.063 ^{4,5}

		EMyO/C	IBEA	MOEA/D	MSOPS	MSOPS2	HypE
15-objectives							
DTLZ1	$\epsilon+$	0.345 ^{5,6}	0.336 ^{4,5,6}	0.136 ^{1,2,4,5,6}	0.424 ^{5,6}	21.212	5.276 ⁵
	HV	0.007 ^{5,6}	0.835 ^{1,4,5,6}	0.918 ^{1,4,5,6}	0.001 ^{5,6}	0	0
	IGD	0.586 ^{4,5,6}	0.321 ^{4,5,6}	0.21 ^{1,2,4,5,6}	0.775 ^{5,6}	35.516	7.495 ⁵
DTLZ2	$\epsilon+$	0.31 ^{2,3,5,6}	0.348 ^{5,6}	0.332 ^{2,5,6}	0.314 ^{2,3,5,6}	0.37 ⁶	0.646
	HV	0.966 ^{3,4,5,6}	0.968 ^{1,3,4,5,6}	0.906 ^{5,6}	0.948 ^{3,5,6}	0.811 ⁶	0.515
	IGD	0.609 ^{2,4,6}	0.662 ⁶	0.523 ^{1,2,4,6}	0.613 ^{2,6}	0.533 ^{1,2,4,6}	0.752
DTLZ3	$\epsilon+$	0.396 ^{2,3,4,5,6}	0.667 ^{4,5,6}	0.501 ^{2,4,5,6}	3.117 ^{5,6}	65.966	11.838 ⁵
	HV	0.82 ^{2,3,4,5,6}	0	0.701 ^{2,4,5,6}	0.001 ²	0.001 ²	0.001 ²
	IGD	0.667 ^{2,4,5,6}	0.939 ^{4,5,6}	0.692 ^{2,4,5,6}	4.907 ^{5,6}	76.096	11.691 ⁵
DTLZ4	$\epsilon+$	0.334 ^{3,4,5,6}	0.315 ^{1,3,4,5,6}	0.369 ⁶	0.348 ^{3,5,6}	0.361 ⁶	1.055
	HV	0.978 ^{2,3,4,5,6}	0.973 ^{3,4,5,6}	0.95 ⁶	0.957 ^{3,6}	0.955 ^{3,6}	0
	IGD	0.682 ^{3,4,5,6}	0.651 ^{1,3,4,5,6}	0.695 ⁶	0.703 ⁶	0.698 ⁶	1.275
DTLZ7	$\epsilon+$	0.772 ^{2,3,4,5,6}	3.625 ⁵	0.793 ^{2,4,5,6}	0.829 ^{2,5,6}	14.944	0.841 ^{2,5}
	HV	0.001 ⁵	0.083 ^{1,3,4,5,6}	0.001 ^{1,5}	0.016 ^{1,3,5,6}	0	0.013 ^{1,3,5}
	IGD	1.419 ^{2,4,5}	1.57 ^{4,5}	1.435 ^{2,4,5}	3.333 ⁵	7.372	1.412 ^{2,3,4,5}
20-objectives							
DTLZ1	$\epsilon+$	0.111 ^{2,4,5,6}	0.363 ^{5,6}	0.063 ^{1,2,4,5,6}	0.26 ^{2,5,6}	2.239	1.751
	HV	0.697 ^{4,5,6}	0.802 ^{4,5,6}	0.997 ^{1,2,4,5,6}	0.363 ^{5,6}	0	0
	IGD	0.187 ^{2,4,5,6}	0.351 ^{4,5,6}	0.142 ^{1,2,4,5,6}	0.43 ^{5,6}	3.251	2.77
DTLZ2	$\epsilon+$	0.383 ^{2,6}	0.434 ⁶	0.332 ^{1,2,5,6}	0.337 ^{1,2,5,6}	0.359 ^{1,2,6}	0.638
	HV	0.98 ^{3,4,5,6}	0.981 ^{3,4,5,6}	0.899 ^{5,6}	0.95 ^{3,5,6}	0.813 ⁶	0.559
	IGD	0.7 ^{2,6}	0.781 ⁶	0.607 ^{1,2,4,5,6}	0.65 ^{1,2,6}	0.633 ^{1,2,6}	0.813
DTLZ3	$\epsilon+$	0.407 ^{2,4,5,6}	0.99 ^{4,5,6}	0.379 ^{2,4,5,6}	2.048 ⁵	8.54	3.093 ⁵
	HV	0.975 ^{2,3,4,5,6}	0.016 ^{4,5,6}	0.83 ^{2,4,5,6}	0	0 ⁴	0 ⁵
	IGD	0.702 ^{2,4,5,6}	0.993 ^{4,5,6}	0.688 ^{1,2,4,5,6}	2.816 ⁵	13.802	2.86 ⁵
DTLZ4	$\epsilon+$	0.4 ^{4,5,6}	0.376 ^{1,4,5,6}	0.385 ^{1,4,5,6}	0.406 ^{5,6}	0.426 ⁶	0.927
	HV	0.992 ^{3,4,5,6}	0.987 ^{3,4,5,6}	0.969 ⁶	0.969 ⁶	0.972 ^{3,4,6}	0.01
	IGD	0.793 ^{4,5,6}	0.759 ^{1,3,4,5,6}	0.772 ^{1,4,5,6}	0.798 ^{5,6}	0.807 ⁶	1.173
DTLZ7	$\epsilon+$	0.795 ^{2,3,4,5,6}	10.519	0.805 ^{2,4,5}	0.839 ^{2,5}	11.53	0.803 ^{2,4,5}
	HV	0.001	0.029 ^{1,3,4,5,6}	0.001 ^{1,5}	0.001 ^{1,3,5}	0.001 ¹	0.001 ^{1,3,5}
	IGD	1.827 ^{2,4,5}	3.204 ^{4,5}	1.802 ^{2,4,5}	3.806	3.892	1.765 ^{1,2,3,4,5}

suggest that approximation sets produced by EMyO/C are relatively close to the true Pareto fronts. The HV values for EMyO/C are always greater than zero, suggesting that EMyO/C generates solutions being within the bounds of the Pareto fronts. Although the IGD indicator is non-Pareto compliant, the small values of IGD indicate the closeness to the Pareto front and adequate distributions of approximations obtained by EMyO/C.

From Table 1, it can be seen that EMyO/C dominates the other algorithms regarding the quality indicators on DTLZ1,3,7 problems with up to 10 objectives, besides IBEA and MOEA/D that give better results with respect to the $\epsilon+$ and HV indicators on DTLZ7 in some dimensions. DTLZ1 and DTLZ3 are multimodal problems with the linear and concave Pareto fronts, whereas the main characteristic of DTLZ7 is the disconnected Pareto front. Thus, EMyO/C appears to be capable of dealing with such problem properties in high-dimensional objective spaces. The competitive results for these problems in 15 and 20 dimensions confirm these observations. DTLZ2 and DTLZ4 do not present much difficulties in terms of the convergence. The best indicator values for these problems in dimensions higher than 3 are obtained by IBEA and MOEA/D, apart DTLZ2 with 5 and 10 objectives, being MSOPS the best algorithm with respect to IGD. The superior performance of IBEA regarding the $\epsilon+$ and HV indicators is not surprising, since its selection procedure relies on the concept of ϵ -dominance. On the other hand, MOEA/D uses a set of uniformly distributed weight vectors that contributes to the high selection pressure and the uniform

Table 2. Mean ranks achieved by different algorithms. The superscripts 1, 2, 3, 4, 5, and 6 indicate whether the respective algorithm performs significantly better than EMyO/C, IBEA, MOEA/D, MSOPS, MSOPS2, and HypE, respectively.

Indicator	EMyO/C	IBEA	MOEA/D	MSOPS	MSOPS2	HypE
$\epsilon+$	1.73 ^{4,5,6}	2.63 ^{5,6}	3.10 ^{5,6}	3.50 ⁵	5.27	4.87
HV	2.08 ^{3,4,5,6}	2.10 ^{3,4,5,6}	3.63	3.82	4.67	4.70
IGD	1.73 ^{4,5,6}	2.97 ^{5,6}	2.90 ^{5,6}	3.97	4.77	4.67

distribution of solutions. Nevertheless, EMyO/C provides a highly competitive performance on all the considered problems.

The overall performance of the algorithms is compared by calculating ranks on each problem with respect to the quality indicators. It should be noted that testing DTLZ1-4,7 problems in 6 different dimensions there is a total of 30 distinct problems. Table 2 presents the mean ranks and statistical comparison. From the table, it can be seen that EMyO/C has the best mean ranks regarding all three indicators, though no difference is detected between EMyO/C and IBEA, as well as there is no difference between EMyO/C and MOEA/D regarding the $\epsilon+$ and HV indicators. These results emphasize the competitiveness of the proposed approach.

3.4 Selection on Different Shapes

We also investigate a generalized EMyO/C, which consists in controlling the shape of \mathcal{F}_l for performing the clustering. It can be defined by the following transformation:

$$\bar{f}_i = f_i^p \quad \forall i \in \{1, \dots, m\}, \quad (8)$$

where $p \in (0, \text{inf})$ is a parameter controlling the shape, f_i is the value obtained in (7), and $\bar{f}_i \in (0, 1)$ is the resulting value. For $p > 1$ solutions in \mathcal{F}_l are projected onto a convex shape, for $p < 1$ solutions in \mathcal{F}_l are projected onto a concave shape. According to p , we define three variants of EMyO/C:

1. EMyO/C-linear ($p = 1$).
2. EMyO/C-convex ($p = 2$).
3. EMyO/C-concave ($p = 0.5$).

We run these three variants with the aforementioned settings, including an EMO algorithm referred as EMO/C. The difference between EMyO/C and EMO/C is that the latter performs clustering on the original objective vectors.

Figure 1 shows the graphical representation of the median values of the quality indicators obtained by the EMyO/C variants and EMO/C on the benchmark functions with varying dimensions. From the figure, it can be seen that EMO/C performs poorly on high-dimensional problems, having zero hypervolume and large values of the $\epsilon+$ and IGD indicators. All of the EMyO/C variants have quite similar performance on DTLZ2,4,7. However, the three variants perform differently on multimodal problems. EMyO/C-concave works better on DTLZ1,

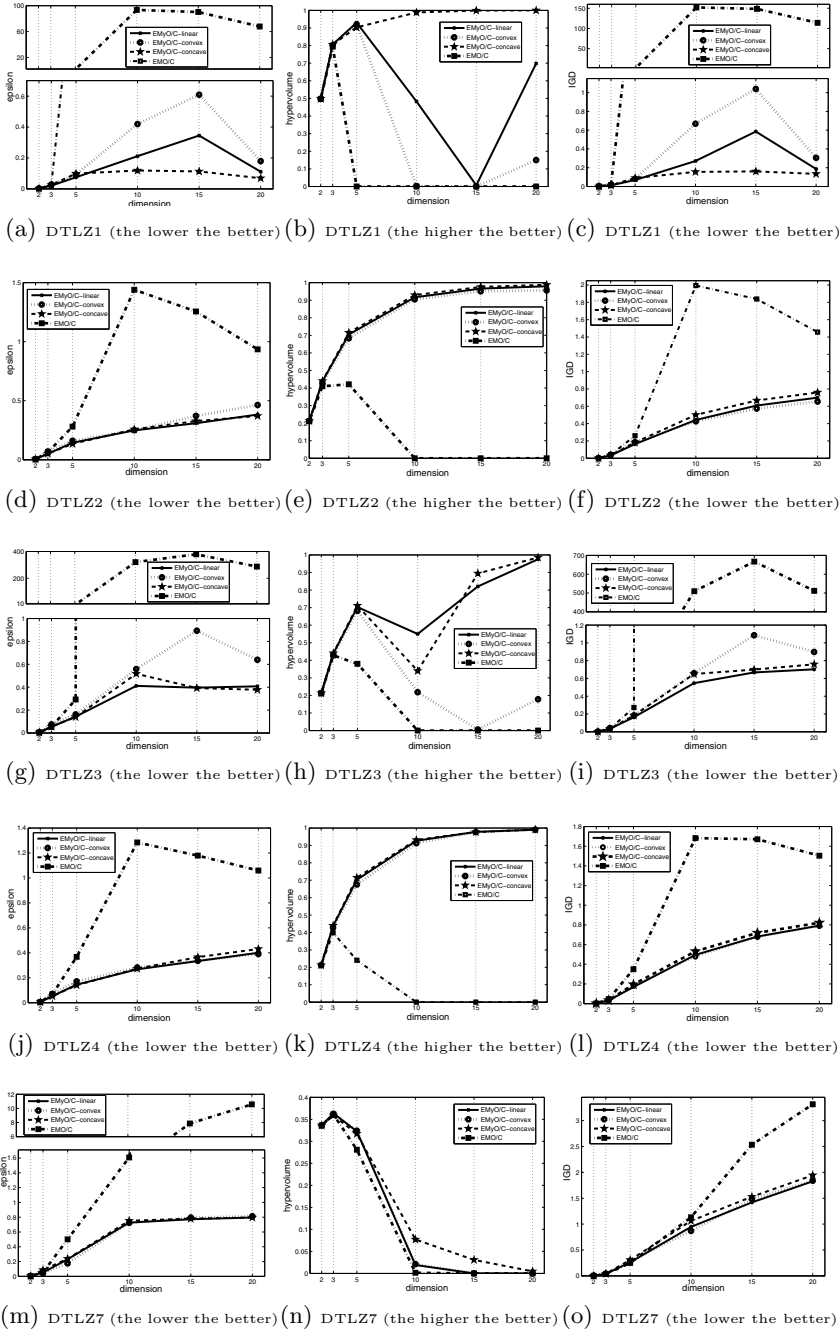


Fig. 1. Performance comparison of EMYO/C-linear, EMYO/C-convex, EMYO/C-concave, and EMO/C on the DTLZ1-4,7 test problems. The plots present the median values of the epsilon (left-hand side), hypervolume (center), and IGD (right-hand side) indicators over 30 runs.

Table 3. Mean ranks achieved by the EMyO/C variants and EMO/C. The superscripts 1, 2, 3, and 4 indicate whether the respective algorithm performs significantly better than EMyO/C-linear, EMyO/C-convex, EMyO/C-concave, and EMO/C, respectively.

Indicator	EMyO/C-linear	EMyO/C-convex	EMyO/C-concave	EMO/C
$\epsilon+$	1.63 ^{2,4}	2.90	2.07 ⁴	3.40
HV	1.87 ^{2,4}	2.92 ⁴	1.50 ^{2,4}	3.72
IGD	1.47 ^{2,3,4}	2.57 ⁴	2.43 ⁴	3.53

whereas EMyO/C-linear and EMyO/C-concave produce different performance in different dimensions. The three variants perform better on DTLZ1 with $m = 20$ than with $m = 15$ due to the smaller number of distance parameters in the former. The obtained results reveal that performing the clustering on different shapes not only affects the distribution of solutions but the convergence and entire performance of the algorithm. Thus, controlling the parameter p can be beneficial for search.

Finally, Table 3 presents the mean ranks and statistical comparison for the performed experiments. It can be seen that the EMyO/C variants are statistically better than EMO/C regarding all three indicators, except for EMyO/C-convex concerning the $\epsilon+$ indicator. EMyO/C-linear gives the best results with respect to the $\epsilon+$ and IGD indicators, whereas EMyO/C-concave performs the best with regard to the HV indicator.

4 Conclusions

In this paper, we proposed a clustering-based selection scheme to guide the search in high-dimensional objective spaces. The experimental results obtained on problems with up to 20 dimensions reveal that the proposed scheme is capable of dealing with many-objective problems, producing a highly competitive performance when compared with the state-of-the-art algorithms. Furthermore, we discussed different variants of the proposed approach, showing their advantages and the relevance of the proposed approach.

As future work, we intend to extend the proposed selection to the domain of GA-based algorithms, developing an effective parent selection mechanism. Further, the self-adaptation of the parameter controlling the shape is another promising direction, as well as calculating the distances in different spaces can bring new opportunities.

Acknowledgements. This work has been supported by FCT – Fundação para a Ciência e Tecnologia in the scope of the project: PEst-OE/EEI/UI0319/2014.

References

1. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008 (2008)

2. Denysiuk, R., Costa, L., Espírito Santo, I.: Many-objective optimization using differential evolution with variable-wise mutation restriction. In: Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO 2013, pp. 591–598 (2013)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
4. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiño, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
5. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation* 13(2), 284–302 (2009)
6. Hughes, E.J.: Multiple single objective Pareto sampling. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003, pp. 2678–2684 (2003)
7. Hughes, E.J.: MSOPS-II: A general-purpose many-objective optimiser. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, pp. 3944–3951 (2007)
8. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation* 19(1), 45–76 (2011)
9. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. Technical Report 112, Swiss Federal Institute of Technology, Zurich, Switzerland (2001)
10. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7(2), 117–132 (2003)
11. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–304. Springer, Heidelberg (1998)
12. Bosman, P.A.N., Thierens, D.: The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 7(2), 174–188 (2003)
13. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC 2005 special session on real parameter optimization. *Journal of Heuristics* 15(6), 617–644 (2009)
14. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – A platform and programming language independent interface for search algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
15. Durillo, J.J., Nebro, A.J.: jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42(10), 760–771 (2011)

On the Impact of Multiobjective Scalarizing Functions

Bilel Derbel^{1,2}, Dimo Brockhoff¹, Arnaud Liefooghe^{1,2}, and Sébastien Verel³

¹ Inria Lille - Nord Europe, DOLPHIN project-team, France

² Université Lille 1, LIFL, UMR CNRS 8022, France

³ Université du Littoral Côte d'Opale, LISIC, France

Abstract. Recently, there has been a renewed interest in decomposition-based approaches for evolutionary multiobjective optimization. However, the impact of the choice of the underlying scalarizing function(s) is still far from being well understood. In this paper, we investigate the behavior of different scalarizing functions and their parameters. We thereby abstract firstly from any specific algorithm and only consider the difficulty of the single scalarized problems in terms of the search ability of a $(1 + \lambda)$ -EA on biobjective NK-landscapes. Secondly, combining the outcomes of independent single-objective runs allows for more general statements on set-based performance measures. Finally, we investigate the correlation between the opening angle of the scalarizing function's underlying contour lines and the position of the final solution in the objective space. Our analysis is of fundamental nature and sheds more light on the key characteristics of multiobjective scalarizing functions.

1 Introduction

Multiobjective optimization problems occur frequently in practice and evolutionary multiobjective optimization (EMO) algorithms have been shown to be well-applicable for them—especially if the problem under study is nonlinear and/or derivatives of the objective functions are not available or meaningless. Besides the broad class of Pareto-dominance based algorithms such as NSGA-II or SPEA2, a recent interest in the so-called *decomposition-based algorithms* can be observed. Those decompose the multi-objective problem into a set of single-objective, ‘scalarized’ optimization problems. Examples of such algorithms include MSOPS [1], MOEA/D [2], and their many variants. We refer to [3] for a recent overview on the topic. The main idea behind those algorithms is to define a set of (desired) search directions in objective space and to specify the scalarizing functions corresponding to these directions. The scalarizing functions can then be solved independently (such as in the case of MSOPS), or in a dependent manner (like in MOEA/D where the recombination and selection operators are allowed to use information from the solutions maintained in neighboring search directions).

Many different scalarizing functions have been proposed in the literature, see e.g. [4] for an overview. Well-known examples are the weighted sum and the (augmented) weighted Chebychev functions, where the latter has an inherent parameter that controls the shape of the lines of equal function values in objective space. Especially with respect to decomposition-based EMO algorithms, it has been reported that the choice of the scalarizing function and their parameters has an impact on the search process [3]. Moreover, it has been noted that adapting the scalarizing function's parameters during the search can allow improvement over having a constant set of scalarizing functions [5]. Although several studies on the impact of the scalarizing function have been

conducted in recent years, e.g. [6], to the best of our knowledge, all of them investigate it on a concrete EMO algorithm and on the quality of the resulting solution sets when more than one scalarizing function is optimized (typically as mentioned above, in a dependent manner). Thereby, the focus is not in understanding why those performance differences occur but rather in observing them and trying to improve the global algorithm. However, we believe that it is more important to first understand thoroughly the impact of the choice of the scalarizing function for a *single* search direction before analyzing more complicated algorithms such as MOEA/D-like approaches with specific neighboring structures, recombination, and selection operators. In this paper, we fundamentally investigate the impact of the choice of the scalarizing functions and their parameters on the search performance, independently of any known EMO algorithm. Instead, we consider one of the most simple single-objective scalarizing search algorithms, i.e., a $(1 + \lambda)$ -EA with standard bit mutation, as an example of a local search algorithm that optimizes a single scalarizing function, corresponding to a single search direction in the objective space. Experiments are conducted on well-understood bi-objective ρ MNK-landscapes.

More concretely, we look experimentally at the impact of the parameters of a generalized scalarizing function (which covers the special cases of the weighted sum and augmented Chebychev scalarizing functions) in terms of the position (angle/direction) reached by the final points, as well as their quality with respect to the Chebychev function. We then consider how the opening of the cones that describe the lines of equal scalarizing function values can provide a theoretical explanation for the impact of the final position of the obtained solutions in objective space. We also investigate the resulting *set quality* in terms of hypervolume and ε -indicator if several scalarizing $(1 + \lambda)$ -EAs are run independently for different search directions in the objective space. Finally, we conclude our findings with a comprehensive discussion of promising research lines.

2 Scalarizing Functions

We consider the maximization of two objectives f_1, f_2 that map search points $x \in X$ to an objective vector $f(x) = (f_1(x), f_2(x)) = (z_1, z_2)$ in the so-called objective space $f(X)$. A solution x is called *dominated* by another solution y if $f_1(y) \geq f_1(x)$, $f_2(y) \geq f_2(x)$, and for at least one i , $f_i(y) > f_i(x)$ holds. The set of all solutions, not dominated by any other, is called Pareto set and its image Pareto front.

Many ways of *decomposing* a multiobjective optimization problem into a (set of) single-objective *scalarizing functions* exist, including the prominent examples of *weighted sum* (WS), *weighted Chebychev* (T), or *augmented weighted Chebychev* (S_{aug}) [4]. For most of them, theoretical results, especially about which Pareto-optimal solutions are attainable, exist [4,7] but they are typically of too general nature to allow for statements on the actual search performance of (stochastic) optimization algorithms. Instead, we are here not interested in any particular scalarizing function, but rather in understanding which general properties of them influence the search behavior of EMO algorithms. We argue by means of experimental investigations that it is not the actual choice of the scalarizing function or their parameters that makes the difference in terms of performance, but rather the general properties of the resulting lines of equal function

Table 1. Overview of the considered scalarizing functions, and the corresponding angles of the lines of equal function values with the standard Pareto dominance cone

scalar function	parameters in S_{gen}	opening angles	reference
$WS(z) = w_1 \bar{z}_1 - z_1 + w_2 \bar{z}_2 - z_2 $	$\alpha = 0, \varepsilon = 1$	$\theta_1 = \arctan\left(-\frac{w_1}{w_2}\right)$ $\theta_2 = \frac{\pi}{2} + \arctan\left(\frac{w_1}{w_2}\right)$	[4, Eq. 3.1.1]
$T(z) = \max\{\lambda_1 \bar{z}_1 - z_1 , \lambda_2 \bar{z}_2 - z_2 \}$	$\alpha = 1, \varepsilon = 0$	$\theta_1 = 0$ $\theta_2 = \pi/2$	[4, Eq. 3.4.2]
$S_{\text{aug}}(z) = T(z) + \varepsilon(\bar{z}_1 - z_1 + \bar{z}_2 - z_2)$	$\alpha = 1,$ $w_1 = w_2 = 1$	$\theta_1 = \arctan\left(-\frac{\varepsilon}{\lambda_1 + \varepsilon}\right)$ $\theta_2 = \frac{\pi}{2} + \arctan\left(\frac{\varepsilon}{\lambda_2 + \varepsilon}\right)$	[4, Eq. 3.4.5]
$S_{\text{norm}}(z) = (1 - \varepsilon)T(z) + \varepsilon WS(z)$	$\alpha = 1 - \varepsilon,$ $w_i = 1/\lambda_i$	$\theta_1 = \arctan\left(-\frac{\varepsilon w_1}{(1 - \varepsilon)\lambda_2 + \varepsilon w_2}\right)$ $\theta_2 = \frac{\pi}{2} + \arctan\left(\frac{\varepsilon w_1}{(1 - \varepsilon)\lambda_1 + \varepsilon w_1}\right)$	here

values. To this end, we consider the minimization of the following general scalarizing function that covers the special cases of WS^1 , T , and S_{aug} functions:

$$S_{\text{gen}}(z) = \alpha \cdot \max\{\lambda_1 \cdot |\bar{z}_1 - z_1|, \lambda_2 \cdot |\bar{z}_2 - z_2|\} + \varepsilon (w_1 \cdot |\bar{z}_1 - z_1| + w_2 \cdot |\bar{z}_2 - z_2|)$$

where $z = (z_1, z_2)$ is the objective vector of a feasible solution, $\bar{z} = (\bar{z}_1, \bar{z}_2)$ a utopian point, $\lambda_1, \lambda_2, w_1$, and $w_2 > 0$ scalar weighting coefficients indicating a search direction in objective space, and $\alpha \geq 0$ and $\varepsilon \geq 0$ parameters to be fixed. For more details about the mentioned scalarizing functions and their relationship, we refer to Table 1.

In the following, we also consider a case of S_{gen} that combines WS and T with a single parameter ε : the normalized $S_{\text{norm}}(z) = (1 - \varepsilon)T(z) + \varepsilon WS(z)$ where $\alpha = 1 - \varepsilon$ and $\varepsilon \in [0, 1]$. For optimizing in a given search direction (d_1, d_2) in objective space, we follow [1,8] and set $\lambda_i = 1/d_i$.² In addition, we refer to the direction angle as $\delta = \arctan(d_1/d_2)$. For the case of S_{norm} , we furthermore choose $w_1 = \cos(\delta)$ and $w_2 = \sin(\delta)$ (thus, $w_1^2 + w_2^2 = 1$) for the weighted sum part in order to normalize the search directions in objective space uniformly w.r.t. their *angles*. Though, in many textbooks you can find statements like “ ε has to be chosen small (enough)”, we do not make such an assumption but want to understand which influence ε has on the finally obtained solutions and how it introduces a trade-off between the Chebychev approach and a weighted sum. For the question of how small ε should be chosen to find all Pareto-optimal solutions in exact biobjective discrete optimization, we refer to [9].

As mentioned above, one important property of a scalarizing function turns out to be the shape of its sets of equal function values, which are known for the WS , T , and S_{aug} functions [4]. However, no description of the equi-function-value lines for the general scalarizing function S_{gen} has been given so far. We think that it is necessary to state those opening angles explicitly in order to gain a deeper intuitive understanding of the above scalarizing approaches and related concepts such as the R2 indicator [8] or more complicated scalarizing algorithms such as MOEA/D [2]. Moreover, it allows us to investigate how a linear combination of weighted sum and Chebychev functions affect the search behavior of decomposition-based algorithms. The following proposition, proven in the accompanying report [10], states these opening angles θ_i between the equi-utility lines and the f_1 -axis, see also Fig. 2 for some examples.

¹ Contrary to the standard literature, our formalization assumes minimization and we therefore have included the utopian point \bar{z} that is typically assumed to be $\bar{z} = (0, 0)$ for minimization.

² The pathologic cases of directions parallel to the coordinates are left out to increase readability.

Table 2. Parameter setting

scalarizing functions	ρ MNK-landscapes	$(1 + \lambda)$ -EA
$\bar{z} = (1, 1)$	$\rho \in \{-0.9, -0.8, \dots, 0.0, \dots, 0.9\}$	$\lambda = n$
$\delta = j \cdot 10^{-2} \cdot \frac{\pi}{2}, j \in \llbracket 1, 99 \rrbracket$	$m = 2$	bit-flip rate = $1/n$
$S_{\text{norm}}: \varepsilon = \ell \cdot 10^{-2}; \ell \in \llbracket 0, 100 \rrbracket$	$n = 128$	stopped after
$S_{\text{aug}}: \varepsilon = \ell \cdot 10^{-k}; \ell \in \llbracket 0, 10 \rrbracket; k \in \llbracket -1, 2 \rrbracket$	$k = 4$	n iterations

Proposition 1. *Let \bar{z} be a utopian point, $\lambda_1, \lambda_2, w_1$, and $w_2 > 0$ scalar weighting coefficients, $\alpha \geq 0$ and $\varepsilon \geq 0$, where at least one of the latter two is positive. Then, the polar angles between the equi-utility lines of S_{gen} and the f_1 -axis are $\theta_1 = \arctan(-\frac{\varepsilon w_1}{\alpha \lambda_2 + \varepsilon w_2})$ and $\theta_2 = \frac{\pi}{2} + \arctan(\frac{\varepsilon w_2}{\alpha \lambda_1 + \varepsilon w_1})$.*

3 Experimental Design

This section presents the experimental setting allowing us to analyze the scalarizing approaches introduced above on bi-objective ρ MNK-landscapes. The family of ρ MNK-landscapes constitutes a problem-independent model used for constructing multiobjective multimodal landscapes with objective correlation [11]. A bi-objective ρ MNK-landscape aims at maximizing an objective function vector $f : \{0, 1\}^n \rightarrow [0, 1]^2$. A correlation parameter ρ defines the degree of conflict between the objectives. We investigate a random instance for each parameter combination given in Table 2.

We investigate the two scalarizing functions S_{norm} and S_{aug} of Table 1 with different parameter settings for the weighting coefficient vector and the ε parameter, as reported in Table 2. In particular, the WS (resp. T) function corresponds to S_{norm} with $\varepsilon = 1$ (resp. $\varepsilon = 0$). The set of weighting coefficient *direction angles* δ_j with respect to the f_1 -axis ($j \in \{1, \dots, 99\}$) are uniformly defined with equal distances in the angle space. For both functions, we set $\lambda_1 = 1/\cos(\delta_j)$, and $\lambda_2 = 1/\sin(\delta_j)$. We recall that for S_{norm} , $w_i = 1/\lambda_i$, and for S_{aug} , $w_i = 1$. To evaluate the relative and the joint performance of the considered scalarizing functions, we investigate the dynamics and the performance of a randomized local search, a simple $(1 + \lambda)$ -EA. After initially drawing a random solution, at each iteration, λ offspring solutions are generated by means of an independent bit-flip mutation, where each bit of the parent solution is independently flipped with a rate $1/n$. The solution with the best (minimum) scalarizing function value among parent and offspring is chosen for the next iteration. For each configuration, 30 independent executions are performed. Due to space limitations, we shall only show a representative subset of settings allowing us to state our findings. More exhaustive results can be found in [10].

4 Single Search Behavior

This section is devoted to the study of the optimization paths followed by *single* independent $(1 + \lambda)$ -EA runs for each direction angle δ and parameter ε of a scalarized problem. In particular, we study the final solution sets reached by the $(1 + \lambda)$ -EA in terms of diversity and convergence and give a sound explanation on how the search behaviour is related to the lines of equal function values of the scalarizing functions.

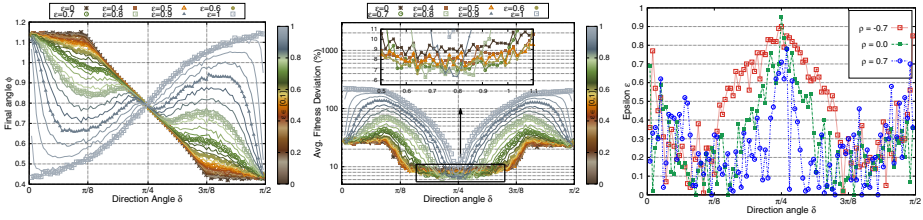


Fig. 1. Left: Angles of final solutions for S_{norm} and $\rho = -0.7$ as a function of δ . Middle: Average deviation to best as a function of weight vector for S_{norm} and $\rho = -0.7$. Right: ε -values providing the smallest deviation for every fixed direction in S_{norm} and $\rho \in \{-0.7, 0.0, 0.7\}$.

4.1 Diversity: Final Angle

In Fig. 1 (Left), we examine the average *angle* of the *final* solution reached by the algorithm with respect to the f_1 -axis using S_{norm} . The final angle of solution x is defined as $\phi(x) = \arctan(f_2(x)/f_1(x))$. It informs about the actual *direction* followed by the search process. We can see that the final solutions are in symmetric positions with respect to direction angle $\pi/4$. This is coherent with the symmetric nature of ρ MNK-landscapes [11]. For WS ($\varepsilon = 1$), every single direction angle infers a different final angle. For T ($\varepsilon = 0$), the extreme direction angles end up reaching ‘similar’ regions of the objective space. These regions actually correspond to the lexicographically optimal points of the Pareto front, which is because of the choice of the utopian point that lies beyond them. Without surprise, we can also see that T and WS do not always allow to approach the same parts of the Pareto front when using the same direction angle.

When varying ε for a fixed δ , the search process is able to span a whole range of positions that are achieved by either T or WS but for variable δ values. Actually, when considering the direction angle being in the middle (i.e. $\delta \approx \pi/4$), the choice of ε does not substantially impact the search direction—because T and WS do allow to move to similar regions in this case. However, as the direction angle goes away from the middle, the influence of ε grows significantly; and the search direction is drifting in a whole range of values. This indicates that the choice of δ is not the only feature that determines the final angle but also the choice of ε highly matters: For some specific ε -values, the direction angles allow to distribute final angles fairly between the two lexicographically optimal points of the Pareto front—in the sense that each direction angle is inferring a different final angle, just like what we observe for WS. For some other ε -values, however, it may happen that the final angles are similar for two different direction angles. In particular, this is the case for large ε -values in S_{norm} , for which WS has more impact than T. We remark that equivalent conclusions can be drawn when examining S_{aug} , which we do not detail here due to lack of space.

The distribution of final directions is tightly related to the diversity of solutions computed by different independent single $(1+\lambda)$ -EAs. As it will be discussed later, this is of crucial importance from a multiobjective standpoint, since diversity in the objective space is crucial to approach different parts of the Pareto front.

4.2 Convergence: Relative Deviation to Best

In the following, we examine the impact of the scalarizing function parameters on the performance of the $(1 + \lambda)$ -EA in terms of convergence to the Pareto front. For that purpose, we compute, for every direction angle δ , the best-found objective vector $z_{\delta, T}^*$ corresponding to the best (minimum) fitness value with respect to T , over all experimented parameter combinations and over all simulations we investigated. For both functions S_{norm} and S_{aug} , we consider the final objective vector z obtained for every direction angle δ and every ε -value. We then compute the relative deviation of z with respect to $z_{\delta, T}^*$, which we define as follows: $\Delta(z) = (T(z) - T(z_{\delta, T}^*)) / T(z_{\delta, T}^*)$. Notice that this relative deviation factor is computed with respect to the T function, which is to be viewed as a reference measure of solution quality. This value actually informs about the performance of the $(1 + \lambda)$ -EA for a fixed direction angle, but variable ε -values.

In Fig. 1 (Middle), we show the average relative deviation to best as a function of direction angles (δ) for different ε -values. To understand the obtained results, one has to keep in mind the results discussed in the previous section concerning the final angles inferred by a given parameter setting. In particular, since WS and T do not infer similar final angles, the final computed solutions lay in different regions of the objective space. Also, for the extreme direction angles, different ranges of ε imply different final angles. Thus, it is with no surprise that the average relative deviation to best can be substantial in such settings. However, the situation is different when considering direction angles in the middle ($\delta \approx \pi/4$). In fact, we observe that for such a configuration, the ε -value does *not* have a substantial effect on final angles, i.e., final solutions lie in similar regions of the objective space. Hence, one may expect that the search process has also the same performance in terms of average deviation to best. This is actually *not* the case since we can observe that the value of ε has a significant impact on the relative deviation for the non-extreme direction angles. To better illustrate this observation, we show, in Fig. 1 (Right), the ε -value providing the minimum average relative deviation to best as a function of every direction angle. We clearly see that the best performances of the $(1 + \lambda)$ -EA for different direction angles are not obtained with the same ε -value.

4.3 Understanding the Impact of the Opening Angle

In this section, we argue that the dynamics of the search process observed previously is rather independent of the scalarizing function under consideration or its parameters. Instead, we show that the search process is guided by the positioning of the lines of equal function values in the objective space—described by the opening angle, i.e., the angle between the line of equal function values and the f_1 -axis (cf. Proposition 1).

Fig. 2 shows three typical exemplary executions of the $(1 + \lambda)$ -EA in the objective space for different parameter settings. The typical initial solution maps around the point $z = (0.5, 0.5)$ in the objective space, which is the average objective vector for a random solution of ρ MNK-landscapes. The evolution of the current solution can be explained by the combination of two effects. The first one is given by the independent bit-flip mutation operator, that produces more offspring in a particular direction compared to the other ones, due to the underlying characteristics of the ρ MNK-landscape under consideration. The second one is given by the lines of equal function values, i.e., the current

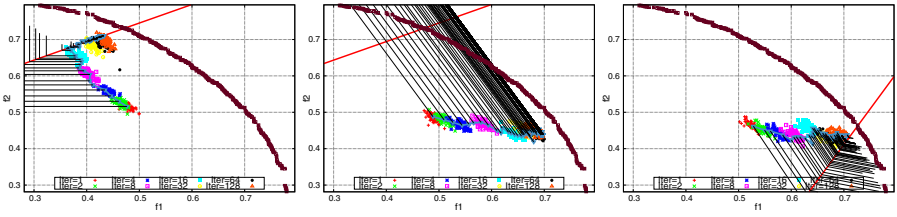


Fig. 2. Exemplary runs of the $(1 + \lambda)$ -EA for different direction angles δ (straight line) and different ϵ -values ($S_{\text{norm}}, \rho = -0.7$). Shown are the best known Pareto front approximation, the offspring at some selected generations, the evolution of the parent, and the lines of equal function values. Left: $\epsilon = 0, \delta = \frac{3}{10} \cdot \frac{\pi}{2}$. Middle: $\epsilon = 1, \delta = \frac{3}{10} \cdot \frac{\pi}{2}$. Right: $\epsilon = 0.6, \delta = \frac{7}{10} \cdot \frac{\pi}{2}$.

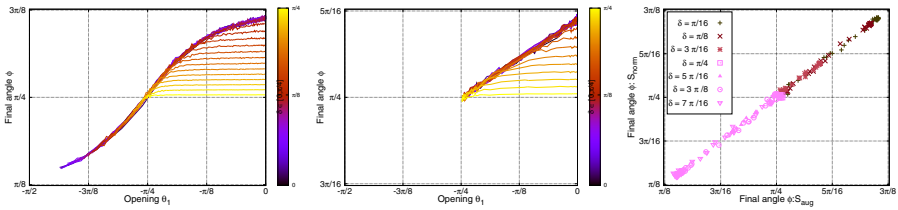


Fig. 3. Left (resp. Middle): scatter plots showing final angle $\phi(\epsilon)$ and opening $\theta_1(\epsilon)$ for $\rho = -0.7$ and S_{norm} (resp. S_{aug}). Every color is for a fixed δ and variable ϵ . Right: Scatter plot showing $(\phi(S_{\text{norm}}), \phi(S_{\text{aug}}))$.

solution moves perpendicular to the iso-fitness lines, following the gradient direction in the objective space. We can remark that the search process is mainly guided by the lower part of the cones of equal function values when the direction is above the initial solution, and *vice versa*. When the direction angle δ is smaller (resp. larger) than $\pi/4$, the dynamics of the search process are better captured by the opening angle θ_1 (resp. θ_2), defined between the equi-fitness lines and the f_1 -axis. Geometrically, the optimal solution with respect to a scalarizing function should correspond to the intersection of one of the ‘highest’ lines of equal fitness values in the gradient direction and the feasible region of the objective space. Although the above description is mainly intuitive, a more detailed analysis can support this general idea.

Let us focus on the influence of the opening angle θ_1 when the direction angle δ is smaller than $\pi/4$ (similar results hold for $\delta > \pi/4$ and θ_2). Fig. 3 shows the scatter plots of the final angle ϕ as a function of the opening angle θ_1 for different direction angles $\delta \in [0, \pi/4]$. A scatter plot gives a set of values $(\theta_1(\epsilon), \phi(\epsilon))$ for the ϵ -values under study. From Proposition 1, for a given direction angle δ , the opening angle θ_1 belongs to the interval $[\delta - \pi/2, 0]$ for S_{norm} , and to the interval $[-\pi/4, 0]$ for S_{aug} . Independently of the scalarizing function, when the direction angle is between 0 and around $3\pi/16$ (blue color), the value of ϕ is highly correlated with the opening angle θ_1 . For such directions, a simple linear regression confirms this observation and allows us to explain the relation between the opening angle and the final angle by means of the following approximate equation: $\phi \approx (c + \pi/4) + c \cdot \theta_1$, such that c equals 0.05, 0.2, and 0.4 for $\rho = -0.7, 0$, and 0.7 respectively. We emphasize that this is independent

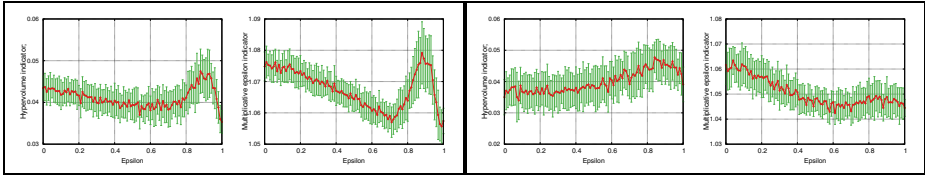


Fig. 4. Column 1 and 3 (resp. 2 and 4) depict the hypervolume (resp. epsilon) indicators for scalar function S_{norm} . Left (resp. right): objective correlations $\rho = -0.7$ (resp. $\rho = 0.7$).

of the definition of the scalarizing function, and depends mainly on the property of the lines of equal function values. The previous equation tells us that the lines of equal fitness values are guiding the search process following the gradient direction given by the opening angle in the objective space. Fig. 3 (Right) shows that the obtained final angles are equivalent when the opening angle is the same, even for different direction angles and/or scalarizing functions. In fact, we observe that the final angles obtained are very similar for the scalarizing functions S_{norm} and S_{aug} if δ is the same for both functions and the ε -values are chosen in order to have matching opening angles. Whatever the δ - and ε -values, the points are close to the line $y = x$, which shows that independently of the scalarizing function, the final angle is strongly correlated to the opening angle, and not to a particular scalarizing function. Also, the opening of the lines of equal function values have more impact on the dynamics of the search process than the direction angle alone. In this respect, the opening angle should be considered as a key feature to describe and understand the behavior of scalarizing search algorithms.

5 Global Search Behavior

In the previous section, we considered every single $(1 + \lambda)$ -EA separately. However, the goal of a general-purpose decomposition-based algorithm is to compute a set of solutions approximating the whole Pareto front. In this section, we study the quality of the set obtained when combining the solutions computed by different configurations of the scalarizing functions. A natural way to do so is to use the same ε -value for all direction angles. Fig. 4 illustrates the relative performance, in terms of hypervolume difference and multiplicative epsilon indicators [12], when considering such a setting and aggregating the solutions from the different weight vectors. The hypervolume reference point is set to the origin, and the reference set is the best-known approximation for the instance under consideration.

Over all the considered ρ MNK-landscapes, we found that the ε -values minimizing both indicator-values correspond to those that allow to well distribute the final angles among direction angles (cf. Fig. 1) independently of the considered scalarizing function. Some differences can however be observed depending on the considered indicator, especially for the most correlated instances as illustrated in Fig. 4. To explain the difference of optimal ε -values between both indicators, we remark that the lexicographically optimal regions of the Pareto front approximation have a higher impact on the hypervolume indicator value, due to the setting of the reference point. For instance, for $\rho = 0.7$, the smallest ε -values concentrate the final angles to the extreme of the Pareto front,

Table 3. Comparison of WS, T, and non-uniform $S_{\text{norm}}^{\varepsilon^*}$ and $S_{\text{aug}}^{\varepsilon^*}$ configured with ε -values giving the best deviation w.r.t every direction. The number in braces shows the number of other algorithms that statistically outperform the algorithm under consideration w.r.t. a given indicator and a Mann-Whitney signed-rank statistical test with a p -value of 0.05 (the lower, the better).

ρ	Avg. hypervolume difference ($\times 10^{-1}$)				Avg. multiplicative epsilon			
	WS	T	$S_{\text{norm}}^{\varepsilon^*}$	$S_{\text{aug}}^{\varepsilon^*}$	WS	T	$S_{\text{norm}}^{\varepsilon^*}$	$S_{\text{aug}}^{\varepsilon^*}$
-0.7	0.353 (2)	0.434 (3)	0.324 (0)	0.307 (0)	1.057 (0)	1.075 (3)	1.059 (0)	1.057 (0)
0.0	0.418 (2)	0.458 (3)	0.357 (1)	0.322 (0)	1.056 (0)	1.084 (3)	1.062 (1)	1.064 (1)
0.7	0.391 (3)	0.350 (2)	0.303 (0)	0.292 (0)	1.044 (0)	1.062 (3)	1.047 (1)	1.047 (1)

which allows to obtain better results in terms of hypervolume. Contrarily, the epsilon indicator values are better when the final angles are well-distributed around $\pi/4$.

Moreover, WS is found to be in general competitive with respect to other fixed ε -values. This observation might suggest that WS is the best-performing parameter setting, since every different direction angle leads to a different final angle. Nevertheless, the diversity of final angles is not the only criterion that can explain quality. The efficiency of the $(1 + \lambda)$ -EA with respect to the single-objective problem implied by the scalarizing function is also crucial. In Fig. 1, we observe that the ε -value exhibiting the minimal average deviation to best is not necessarily the same for every direction. We also observe that for direction angles in the middle of the weight space, the final angles obtained for different ε -values can end up being very similar. Thus, it might be possible that, by choosing different ε -values for different directions, one can find a configuration for which final solutions are diverse, but also closest to the Pareto front. Indeed, we can observe a significant difference between the non-uniform case where the scalarizing function S_{norm} (or S_{aug}) is configured with an ε providing the best deviation to best for every direction, and the situation where ε is the same for all directions. As shown in Table 3, such non-uniform configurations are both substantially better than T and also competitive compared to WS. We only show the performance of the above non-uniform configuration in order to illustrate how choosing different ε -values can improve the quality of the resulting approximation set. However, this particular non-uniform configuration might not be ‘optimal’. In other words, finding the ‘best’ parameter configuration in a setting where μ independent single $(1 + \lambda)$ -EAs are considered, can itself be formulated as an optimization problem with variables ε and δ ; such that direction angles in the optimal configuration might not necessarily be pairwise different.

6 Open(ing) (Re)search Lines

We presented an extensive empirical study that sheds more light on the impact of scalarizing functions within decomposition-based evolutionary multiobjective optimization. Our results showed that, given a weighting coefficient vector and a relative importance of the weighted sum and the Chebychev term in the function, it is fundamentally the opening of the lines of equal function values that explicitly guides the search towards a specific region of the objective space. When combining multiple scalarizing search processes to compute a whole approximation set, these lines play a crucial role to achieve diversity. While our results are with respect to a rather simple setting where multiple scalarizing search procedures are run independently, they make a fundamental step

towards strengthening the understanding of the properties and dynamics of more complex algorithmic settings. It is our hope that the lessons learnt from our study can highly serve to better tackle the challenges of decomposition-based approaches. They also rise new interesting issues that were hidden by the complex design of well-established algorithms. In the following, we identify a non-exhaustive number of promising research directions that relate directly to our findings.

❶ Improving Existing Algorithms. Eliciting the best configuration to tackle a multiobjective optimization problem by decomposition can highly improve search performance. As we demonstrated, similar regions can be achieved using different parameter settings, and the performance could be enhanced by adopting non-uniform configurations. One research direction would be to investigate how such *non-uniform* configurations perform when plugged into existing approaches. To our best knowledge, there exists no attempt in this direction, and previous investigations did only consider uniform parameters, which do not necessarily guarantee to reach an optimal performance.

❷ Tuning the Opening Angles. Generally speaking, the parameters of existing scalarizing functions can simply be viewed as one specific tool to set up the openings of the lines of equal function values. In this respect, other types of opening angles can be considered without necessarily using a particular scalarizing function. This would offer more flexibility when tuning decomposition-based algorithms, e.g., defining the opening angles without being bound to a fixed closed-form definition, but adaptively, with respect to the current search state. We believe that classical paradigms for on-line and off-line parameter setting are worth to be investigated to tackle this challenging issue.

❸ Variation Operators and Problem-Specific Issues. In our study, we consider the independent bit-flip mutation operator and bi-objective ρ MNK-landscapes. In future work, other problem types and search components should be investigated at the aim of gaining in generality—also towards problems with more than two objectives.

❹ Theoretical Modeling. A challenging issue is to provide a framework, abstracting from problem-specific issues, and allowing us to reason about decomposition-based approaches in a purely theoretical manner. This would enable us to better harness scalarizing approaches and to derive new methodological tools in order to improve our practice of decomposition-based evolutionary multiobjective optimization approaches.

References

1. Hughes, E.J.: Multiple Single Objective Pareto Sampling. In: CEC, pp. 2678–2684 (2003)
2. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE TEC 11(6), 712–731 (2007)
3. Giagkiozis, I., Purshouse, R.C., Fleming, P.J.: Generalized Decomposition. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 428–442. Springer, Heidelberg (2013)
4. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
5. Ishibuchi, H., Sakane, Y., Tsukamoto, N., Nojima, Y.: Adaptation of scalarizing functions in MOEA/D: An adaptive scalarizing function-based multiobjective evolutionary algorithm. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 438–452. Springer, Heidelberg (2009)

6. Ishibuchi, H., Akedo, N., Nojima, Y.: A study on the specification of a scalarizing function in MOEA/D for many-objective knapsack problems. In: LION7, pp. 231–246 (2013)
7. Kaliszewski, I.: Using trade-off information in decision-making algorithms. *Computers & Operations Research* 27(2), 161–182 (2000)
8. Brockhoff, D., Wagner, T., Trautmann, H.: On the Properties of the $R2$ Indicator. In: Genetic and Evolutionary Computation Conference, GECCO 2012, pp. 465–472 (2012)
9. Dächert, K., Gorski, J., Klamroth, K.: An Augmented Weighted Tchebycheff Method With Adaptively Chosen Parameters for Discrete Bicriteria Optimization Problems. *Computers & Operations Research* 39(12), 2929–2943 (2012)
10. Derbel, B., Brockhoff, D., Liefoghe, A., Verel, S.: On the impact of scalarizing functions on evolutionary multiobjective optimization. Research Report RR-8512, INRIA Lille - Nord Europe (March 2014),
<http://hal.inria.fr/docs/00/96/81/45/PDF/RR-8512.pdf>
11. Verel, S., Liefoghe, A., Jourdan, L., Dhaenens, C.: On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives. *Eur. J. Oper. Res.* 227(2), 331–342 (2013)
12. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE TEC* 7(2), 117–132 (2003)

Multi-objective Quadratic Assignment Problem Instances Generator with a Known Optimum Solution

Mădălina M. Drugan

Artificial Intelligence lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
mdrugan@vub.ac.be

Abstract. *Multi-objective quadratic assignment problems* (mQAPs) are NP-hard problems that optimally allocate facilities to locations using a distance matrix and several flow matrices. mQAPs are often used to compare the performance of the multi-objective meta-heuristics. We generate large mQAP instances by combining small size mQAP with known local optimum. We call these instances *composite mQAPs*, and we show that the cost function of these mQAPs is additively decomposable. We give mild conditions for which a composite mQAP instance has known optimum solution. We generate composite mQAP instances using a set of uniform distributions that obey these conditions. Using numerical experiments we show that composite mQAPs are difficult for multi-objective meta-heuristics.

1 Introduction

The Quadratic assignment problem (QAP) models many real-world problems like the computer-aided design in the electronics industry, scheduling, vehicle routing, etc. Intuitively, QAPs can be described as the (optimal) assignment of a number of facilities to a number of locations. In general, QAP instances are NP hard problems, and QAP instances are often included in the benchmarks for testing meta-heuristics [1, 2]. Special cases of QAPs solvable in polynomial time are easy to solve [3]. Meta-heuristic search algorithms based on local search are especially useful for large size QAPs, where exact solutions are difficult to obtain. Furthermore, measuring the performance of meta-heuristics is best done when the optimum solution for the test problem is known.

Generating large size QAPs with known local optimum solutions that are difficult and interesting for exact and stochastic algorithms is a current challenge in the field [4, 5]. The algorithms that generate large and hard single objective QAP instances with known optima [6] are rather elaborated and difficult to generate. Drezner et al [4] propose QAP instances that are difficult to solve with heuristics but easy for exact solvers because of the large amount of 0's in the flow matrix.

Recently, Drugan [5] proposes a single objective QAP instance generator with additively decomposable cost function and known local optimum. Problems with additively decomposable cost functions are considered useful test benchmark for meta-heuristic algorithms that explore the structure of the search space. These QAP instances are difficult for both exact methods, like branch and bound, and for meta-heuristics.

Multi-objective Quadratic Assignment Problems [7] are an extension of QAP with more than two flow matrices. Let us consider N facilities, the $N \times N$ distance

matrix $A = (a_{ij})$, where a_{ij} is the distance between location i and location j . Consider an mQAP with m flow matrices $\mathbf{B} = (B^1, \dots, B^m)$, where $m \leq 2$ and $B^o = (b_{ij}^o)$ and b_{ij}^o represents the k -th flow matrix from facility i to facility j . The goal is to minimise the *cost function* in all objectives o

$$c^o(\pi) = \sum_{i=1}^N \sum_{j=1}^N a_{ij} \cdot b_{\pi_i \pi_j}^o \quad (1)$$

where $1 \leq o \leq m$ and π is a permutation of N facilities and π_i is the i -th element of π . It takes quadratic time to evaluate each of these functions. We consider an mQAP as a tuple (A, B^1, \dots, B^m, s) where s , if known, is the optimum solution.

The Main Contribution. We design a multi-objective QAP instance generator that creates meaningful, i.e. large and difficult to solve, benchmark instances for multi-objective meta-heuristics [2, 8]. Our solution introduced in Section 2 is to aggregate several flow and distance matrices with computable optimum solutions, into a larger mQAP such that the optimum of the resulting mQAP is known, called *composite mQAPs*. These mQAPs have additively decomposable cost functions that are the sum of component mQAP's cost functions plus an extra term corresponding to the cost of the region outside these component mQAPs.

In Section 3, we give mild conditions, e.g. upper and lower bounds for the values in the mQAPs matrices such that the composite mQAP instance has the identity permutation as the global optimum solution. However, to verify the global optimum solution we compute a large number of cost functions equivalent with the number of permutations of the component mQAP instances into the permutation of the composite mQAP instance.

In order to simplify the procedure of generating composite mQAPs with known global optima, we consider uniform distributions which are also used to generate other mQAPs from the literature [2, 7]. In Section 4, the conditions on the upper and lower bounds are easily verifiable, and explicit numerical values are proposed.

Numerical experiments from Section 5 show that the composite mQAPs are difficult to solve with multi-objective meta-heuristic instances [8] when compared with the other mQAPs from literature [7]. We show that the global optimum is difficult to attain and thus composite mQAPs are difficult to solve. Section 6 concludes the paper.

2 Composite Multi-objective QAP Instances Generator

In this section, we design an algorithm that generates *composite mQAP* instances from small size *component mQAP* instances with computable optimum solution. The values in the composite mQAP not assigned yet are also selected to have known optimum value. Thus, there are three optimisation problems in composite mQAPs: i) optimising the component mQAPs, ii) optimising the region outside these components, and iii) a global optimisation problem for the entire mQAP. The pseudo-code for this algorithm is given in Algorithm 1.

The algorithm `generate_composite_mQAP` has as input d component mQAP instances, $(A_k, B_k^1, \dots, B_k^m, \mathcal{I})$, $\forall k \leq d$, with identity permutation \mathcal{I} as optimum solution, where $\forall i \in \{1, \dots, N\}, \mathcal{I}_i = i$. In order to calculate the optimum solution of

Algorithm 1. generate_composite_mQAP

Require: d component mQAP instances $\{(A_1, B_1^1, \dots, B_1^m, \mathcal{I}), \dots, (A_d, B_d^1, \dots, B_d^m, \mathcal{I})\}$
Require: the distributions in the outside region $\mathcal{R}_A, \mathcal{R}_{B^1}, \dots, \mathcal{R}_{B^m}$; the low values distributions $\mathcal{L}_A, \mathcal{L}_{B^1}, \dots, \mathcal{L}_{B^m}$, and the high values distributions $\mathcal{H}_A, \mathcal{H}_{B^1}, \dots, \mathcal{H}_{B^m}$
/ I. Aggregate mQAP instances/**

 Initialise A, B^1, B^2, \dots, B^m with 0s everywhere

for all $k = 1$ to d **do**
for all $i, j = 1$ to n_k **do**
 $t \leftarrow i + \sum_{r=1}^k n_r; p \leftarrow j + \sum_{r=1}^k n_r;$
 $a_{tp} \leftarrow a_{tp} + a_{kij}; b_{tp}^1 \leftarrow b_{tp}^1 + b_{kij}^1; \dots; b_{tp}^m \leftarrow b_{tp}^m + b_{kij}^m;$
end for
end for
/ II. Generate the set of elements in A, B^1, \dots, B^m not assigned yet/**
for all $\alpha\%$ elements $a_{ij} \in \mathcal{R}_A, b_{ij}^1 \in \mathcal{R}_{B^1}, \dots, b_{ij}^m \in \mathcal{R}_{B^m}$ **do**

 Generate $a_{ij} \propto \mathcal{H}_A$, and update the sorted list $\mathcal{R}_A \leftarrow \mathcal{R}_A \cup a_{ij}$

 Generate $b_i^o \propto \mathcal{L}_{B^o}$, and update the sorted list $\mathcal{R}_{B^o} \leftarrow \mathcal{R}_{B^o} \cup b_i^o$, for all $o \leq m$
 $t \leftarrow t + 1$
end for
for all $(1 - \alpha)\%$ elements $a_{ij} \in \mathcal{R}_A, b_{ij}^1 \in \mathcal{R}_{B^1}, \dots, b_{ij}^m \in \mathcal{R}_{B^m}$ **do**

 Generate $a_{ij} \propto \mathcal{L}_A$, and update the sorted list $\mathcal{R}_A \leftarrow \mathcal{R}_A \cup a_{ij}$

 Generate $b_i^o \propto \mathcal{H}_{B^o}$, and update the sorted list $\mathcal{R}_{B^o} \leftarrow \mathcal{R}_{B^o} \cup b_i^o$, for all $1 \leq o \leq m$
 $t \leftarrow t + 1$
end for
for all $r = 1$ to $|\mathcal{R}_A|$ **do**
 $r \leftarrow$ rank of a_{ij} in \mathcal{R}_A
 $b_{ij}^o \leftarrow b_i^o$ with rank $|\mathcal{R}_A| - r$ in \mathcal{R}_{B^o} , for all $o \leq m$
end for
return (A, B^1, \dots, B^m)

component mQAPs, we could, for example, exhaustively enumerate all possible permutations. A straightforward method to transform a component mQAP with an optimum solution s into an mQAP instance with the identity permutation as optimum solution is to rename the facilities.

2.1 Aggregate Component mQAP Instances

For simplicity, we consider that each facility from the composite mQAP corresponds to exactly one facility from a single component mQAP, and, vice-versa, each facility from a component mQAP corresponds to exactly one facility from the composite mQAP. We consider that n_k are the number of facilities of the k -th component mQAP, $(A_k, B_k^1, \dots, B_k^m, \mathcal{I})$. We call the reunion of all component mQAPs the *component region*. Note that the number of facilities N for the newly generated composite mQAP is the sum of the number of facilities of the component mQAP, $N = \sum_{k=1}^d n_k$.

For each pair of facilities in the k -th component mQAP $(i, j) \in A_k$, there is assigned a pair of facilities in the composite mQAP $(t, p) \in A$, where $t \leftarrow i + \sum_{r=1}^k n_r$ and

$p \leftarrow j + \sum_{r=1}^k n_r$. We update the values $a_{tp} \in A$ and $b_{tp}^o \in B^o$ with the corresponding values in $a_{kij} \in A_k$ and $b_{kij}^o \in B_k^o, \forall o \leq m$.

2.2 Filling up the Composite mQAP Instances

Next, we assign the positions in A and B not assigned yet. Let \mathcal{R}_A and \mathcal{R}_{B^o} be ordered sets containing all unassigned values from A , and B^o , respectively. We call these sets, the *outside region* of the corresponding matrices. The elements in the outside region are generated using the rearrangement inequality [9]¹ such that their cost function has the identity permutation as the optimum solution. Informally, the largest values in the o -th flow matrix B^o correspond to the lowest values in the distance matrix A , and the lowest values in B^o correspond to the largest values in A .

The low values distributions \mathcal{L}_A and \mathcal{L}_{B^o} generate the lowest values of A and B^o , respectively. The high values distributions \mathcal{H}_A and \mathcal{H}_{B^o} generate the highest values of A and B^o . We generate $\alpha\%$ unassigned values in A from \mathcal{H}_A and $(1 - \alpha)\%$ from \mathcal{L}_A . Because of the rearrangement inequality, $\alpha\%$ values in each of the flow matrices B^o are generated from \mathcal{L}_{B^o} and $(1 - \alpha)\%$ are generated from \mathcal{H}_{B^o} .

In Algorithm 1, let r be the rank of a_{ij} in \mathcal{R}_A . If a_{ij} is generated from \mathcal{H}_A , then each value b_{ij}^o is generated from \mathcal{L}_{B^o} such that the rank of b_{ij}^o in \mathcal{R}_{B^o} is $|\mathcal{R}_A| - r$. Similarly, if a_{ij} is generated from \mathcal{L}_A , then b_{ij}^o is generated from \mathcal{H}_{B^o} such that the rank of b_{ij}^o in \mathcal{R}_{B^o} is $|\mathcal{R}_A| - r$. Thus, the elements $b_{ij}^1, \dots, b_{ij}^m$ have the same ranking in the outside regions of the corresponding flow matrices, B^1, \dots, B^m .

3 Designing Composite mQAPs with Known Optimum Solution

Cela [3] showed that single QAP instances where all the elements obey the rearrangement inequality are *easy*. This means that if the component mQAPs are degenerated, $n_1 = \dots = n_d = 1$, then the composite mQAP also becomes "easy". Thus, we consider the component mQAPs to be the "difficult" region, and the outside region to be the "easy" region of a composite mQAP. By design, the component mQAPs and the outside region are optimised by the identity permutation. The composite mQAP, in general, is not optimised by the identity permutation.

In this section, we give mild conditions under which the composite mQAP instances have the identity permutation as the optimum solution. We consider that all the elements in the outside region are either smaller or larger than all the elements in the component mQAPs. Accordingly to the rearrangement inequality, if elements are exchanged between the component mQAPs and the outside region, then the cost of the composite mQAP instance increases.

Additively Decomposable Cost Functions for the Composite mQAPs. In the following, we show that the composite mQAP instances have *additively decomposable cost functions* with a residual term representing the cost of the outside region.

¹ Let n variables be generated with any two distributions $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_n\}$ for which $x_1 \leq \dots \leq x_n$ and $y_1 \geq \dots \geq y_n$. The rearrangement inequality states that $\sum_{i=1}^n x_i \cdot y_i \leq \sum_{i=1}^n x_i \cdot y_{\pi_i}$, for all permutations π .

Consider the set $\Pi(N)$ of all permutations of N facilities in the flow matrices. In the permutation group theory, permutations are often written in the cyclic form. If π is a permutation of facilities, we can write it as $\pi = (\pi_1, \dots, \pi_d)$, where π_k is the k -th cycle containing a set of facilities that can be swapped with each other. These cycles are disjoint subsets. We consider d cycles, each cycle contains the facilities of exactly one component mQAP. If there are n_k facilities in the k -th component mQAP, the corresponding cycle is a n_k -cycle. The cost function of the k -th cycle is

$$c_k^o(\pi) = \sum_{i,j,\pi_i,\pi_j} a_{kij} \cdot b_{k\pi_i\pi_j}^o \tag{2}$$

where $k \in \{1, \dots, d\}$, d is the number of component QAPs and a_{kij} is an element of the k -th component QAP. Similarly, $b_{k\pi_i\pi_j}^o$ is an element of the k -th component QAP. By design, the optimal cost for each cycle in each objective is $c_k^o(\mathcal{I}) \leftarrow \min_{\pi} c_k^o(\pi)$.

The cost function of π is now

$$c^o(\pi) = \sum_{i=1}^N \sum_{j=1}^N a_{ij} \cdot b_{\pi_i\pi_j}^o = \sum_{k=1}^d c_k^o(\pi) + R^o(\pi) \tag{3}$$

where $R^o(\pi)$ is a residue defined as the cost in the outside region for the flow matrix o

$$R^o(\pi) = \sum_{a_{ij} \in \mathcal{R}_A, b_{\pi_i\pi_j}^o \in \mathcal{R}_{B^o}} a_{ij} \cdot b_{\pi_i\pi_j}^o \tag{4}$$

Swapping facilities *in* a cycle results in swapping elements in the component mQAP and in the outside region. Swapping facilities *between* cycles results in swapping elements between the component mQAPs and the outside region.

3.1 Setting Up Bounds for the Generating Distributions

Let m_A and m_{B^o} be the smallest element in the component distance matrices A_k , $m_A \leftarrow \min_{k \leq d} \{a_{kij}\}$, and the component flow matrices B_k^o , $\forall o$, $m_{B^o} \leftarrow \min_{k \leq d} \{b_{kij}^o\}$, respectively. Similarly, $M_A \leftarrow \max_{k \leq d} \{a_{kij}\}$ and $M_{B^o} \leftarrow \max_{k \leq d} \{b_{kij}^o\}$. Let ℓ_A and L_A be the lowest and the highest bound for the distribution \mathcal{L}_A , and let ℓ_{B^o} and L_{B^o} be the lowest and the highest bound for \mathcal{L}_{B^o} . Let h_A and H_A be the lowest and the highest bound for \mathcal{H}_A and let h_{B^o} and H_{B^o} be the lowest and the highest bound for \mathcal{H}_{B^o} .

The next proposition sets conditions on the bounds for the composite mQAP with the identity permutation as the optimum solution.

Proposition 1. *Let be $\{(A^k, B_k^1, \dots, B_k^m, \mathcal{I}) \mid k = 1, \dots, d\}$ a set of equal sized mQAP instances with the optimum solution the identity permutation. Algorithm 1 generates a composite mQAP from these component mQAPs. Let following equations hold*

$$\ell_A < L_A < m_A < M_A < h_A < H_A \tag{5}$$

$$\ell_{B^o} < L_{B^o} < m_{B^o} < M_{B^o} < h_{B^o} < H_{B^o} \tag{6}$$

$$\min\{m_A, m_{B^o}\} \cdot (\min\{h_A, h_{B^o}\} + \min\{\ell_A, \ell_{B^o}\}) > M_A \cdot M_{B^o} + \min\{h_A \cdot L_{B^o}, L_A \cdot h_{B^o}\} \tag{7}$$

$$\sum_{k=1}^d (c_k^o(\mathcal{I}) - c_k^o(\pi)) + \sum_{\alpha_{ij} \in \mathcal{R}_A, b_{\pi_i \pi_j}^o \in \mathcal{R}_{B^o}} \alpha_{ij} \cdot (b_{ij}^o - b_{\pi_i \pi_j}^o) < 0 \tag{8}$$

where π any permutation and for all objectives $o \leq m$. Then, the composite mQAP $(A, B^1, \dots, B^m, \mathcal{I})$ has the identity permutation as the optimum solution.

Proof. The proof follows directly from the proof of Proposition 1 from [5]. Intuitively, the set $\Pi(N)$ of all possible permutations is split in three subsets: i) exchange facilities *within* a cycle, ii) cycle that completely switch their facilities with other cycles, and iii) the general case where facilities are switched at random between cycles. The proof considers the difference between the identity permutation and another permutation for all these three cases. \square

In Proposition 1, for Inequality 5 and 6, the rearrangement inequality holds. From Inequality 7 and the rearrangement inequality, we have that a permutation where facilities are swapped between the outside and the component region has a higher cost than a permutation where solutions are swapped *in* the composite or *in* the outside region. The condition in Inequality 7 can be fulfilled by setting the bounds for the distributions \mathcal{H}_A and \mathcal{H}_{B^o} high enough.

Inequality 8 states that if swapping elements in the outside region generates more variance than swapping elements in the component mQAPs, then the identity permutation is the global minimum for the subset of permutations where cycles are completely swapped. To decide if the generated composite mQAP has the identity permutation as optimum solution, we need $d!$ evaluations of Inequality 8 corresponding to all combinations of the component mQAPs on the diagonal of the composite mQAP.

4 A Practical Composite mQAP Instance Generator

In this section, we generate composite mQAP instances to fulfil the conditions from Proposition 1. The current mQAP instance generators [2, 7] use uniform distributions to generate mQAPs. Thus, we also use uniform distributions to generate composite mQAPs. Note that even though component mQAPs and the elements in the outside region are generated by uniform random distributions, the values of the corresponding composite mQAP instances are not generated by a uniform random distribution.

An uniform random distribution \mathcal{D} generates all the component mQAPs. Let \mathcal{L} and \mathcal{H} be the uniform independent distributions generating the outside region of the distance matrix A and the flow matrices \mathbf{B} .

We study the relationship between the inequalities from Proposition 1 on the bounds for the uniform distributions. Let the two terms from Inequality 8 be denoted as the variance of the composite region and of the outside region

$$\Delta_C = \sum_{k=1}^d c^k(\mathcal{I}) - c^k(\pi), \quad \Delta_{\mathcal{O}} = R^o(\mathcal{I}) - R^o(\pi)$$

We explicitly compute the values of Δ_C and $\Delta_{\mathcal{O}}$. If $\Delta_{\mathcal{O}} + \Delta_C$ is non-negative, the identity permutation is the global optimum solution.

Consider that there are $L - \ell + 1$ values in \mathcal{L} , $\ell = s_0, s_1, \dots, s_{L-\ell} = L$, and $H - h + 1$ uniformly generated values in \mathcal{H} , ($h = t_0, t_1, \dots, t_{H-h} = H$). Let's assume that $L - \ell = H - h$. With a perfect random generator, in any row and column of mQAPs' matrices values of \mathcal{L} and of \mathcal{H} are equally represented.

The Variance in the Outside Region. Assuming that all the values of the distributions \mathcal{L} and \mathcal{H} are uniformly distributed, the cost of the outside region has the approximative value of

$$R^o(\mathcal{I}) = \sum_{a_{ij} \in \mathcal{R}_A} a_{ij} \cdot b_{ij}^o \approx \frac{|\mathcal{R}_A|}{H - h + 1} \cdot \left(\sum_{i=0}^{L-\ell} s_i \cdot t_{H-h-i} \right) \tag{9}$$

When $\alpha = 0.5$, the elements in the flow and distance matrices are equally generated from low and high distributions. The swapped elements are randomly distributed in the corresponding matrices and, thus, the cost of the outside region in each objective o is upper bounded by

$$R^o(\pi) \leq \frac{|\mathcal{R}_A|}{L - \ell + H - h + 2} \cdot \left(\sum_{i=0}^{L-\ell} s_i + \sum_{j=0}^{H-h} t_j \right)^2$$

For a permutation π , let assume that $(1 - p) \cdot |\mathcal{R}_A|$ percent of the outside region is optimised and the remaining $p \cdot |\mathcal{R}_A|$ percent of the outside region is uniform randomly positioned in the matrix. Then the cost of the outside region in each objective o is

$$R^o(\pi) \approx \frac{(1 - p) \cdot |\mathcal{R}_A|}{L - \ell + 1} \cdot \left(\sum_{i=0}^{L-\ell} s_i \cdot t_{L-\ell-i} \right) + \frac{p \cdot |\mathcal{R}_A| \cdot \left(\sum_{i=0}^{L-\ell} s_i + \sum_{j=0}^{H-h} t_j \right)^2}{L - \ell + H - h + 2}$$

Given a certain value for p , the variance is the outside region is

$$\Delta_{\mathcal{O}}^o \approx \frac{p \cdot |\mathcal{R}_A|}{L - \ell + 1} \cdot \left(\sum_{i=0}^{L-\ell} s_i \cdot t_{L-\ell-i} \right) - \frac{p \cdot |\mathcal{R}_A| \cdot \left(\sum_{i=0}^{L-\ell} s_i + \sum_{j=0}^{H-h} t_j \right)^2}{L - \ell + H - h + 2} \tag{10}$$

The variance in the component mQAPs. The minimum cost of all d component mQAPs is approximatively equal because all the values are generated from the same uniform distribution. This cost could be increased by the imperfection of the random generator, and the limited size of the component mQAP. Consider that there are $M - m + 1$ values in \mathcal{D} , such that $(m = v_0, v_1 \dots, v_{M-m} = M)$. Let $N^2 - |\mathcal{R}_A| = d \cdot n \cdot (n - 1)$ be the total number of elements in the component mQAPs. Following the same line of reasoning, the maximum variance is

$$\Delta_C < \sum_{k=1}^d c^k(\mathcal{I}) - \frac{N^2 - |\mathcal{R}_A|}{(M - m + 1)^2} \left(\sum_{i=0}^{M-m} v_i \right)^2 \tag{11}$$

The elements of the component matrices are uniform randomly generated and positioned. Thus, when the component mQAPs are optimised and $N \rightarrow \infty$, we have

$$\sum_{k=1}^d c^k(\mathcal{I}) \approx \frac{N^2 - |\mathcal{R}_A|}{(M - m + 1)^2} \left(\sum_{i=0}^{M-m} v_i \right)^2 \tag{12}$$

and $\Delta_C \rightarrow 0$.

The Variance in the Composite mQAPs. Note that if $N \rightarrow \infty$, then Δ_C is approaching 0, and Δ_O has a negative value, $\Delta_O < 0$, and the identity permutation is the optimal solution. This concludes our reasoning.

4.1 An Example

We choose the bounds for the composite mQAPs to be the same with the bounds for the uniform randomly generated mQAPs from [2, 7] with the purpose of comparing the mQAP instances. These bounds are also set to cover a large number of values between 1 and 99, the same bounds as the randomly generated mQAP instances. Let's consider the following numerical values: i) $m = 21$ and $M = 40$, ii) $h = 80$ and $H = 99$, and iii) $\ell = 1$ and $L = 20$. Thus $L - \ell = H - h = 20$. Let $n = 8$ be the number of facilities in component mQAPs, where $d \geq 2$. Further, $s_i = i$ and $t_i = i + 80$, where $i \in \{1, \dots, 20\}$.

If $d = 2, 3, \dots$, then $|\mathcal{R}_A| = 128, 384, 768, \dots$ Using Equation 9, the cost of the outside region is $R^o(\mathcal{I}) = |\mathcal{R}_A| \cdot \frac{1 \cdot 99 + \dots + 20 \cdot 80}{20} = |\mathcal{R}_A| \cdot 906.5$. From Equation 10, we have that $\Delta_O \approx p \cdot |\mathcal{R}_A| \cdot 906.5 - p \cdot |\mathcal{R}_A| \cdot 6703.2 \approx -p \cdot |\mathcal{R}_A| \cdot 5807.7$. Note that the second term it is negative and dominates Δ_O . From $N^2 - |\mathcal{R}_A| = 112, 168, 224, \dots$ and Equation 11, we have that $\Delta_C < \sum_{k=1}^d c^k(\mathcal{I}) - (N^2 - |\mathcal{R}_A|) \cdot 29241$. From Equation 12, we have that $\Delta_C \rightarrow 0$, and thus the identity permutation is the optimum solution for all the composite mQAPs with these uniform distributions. Note that the condition from Inequality 7 was relaxed for this numerical example.

5 Difficulty of mQAP Instances

Our goal is to generate instances that are difficult to solve with local search. We propose to use as difficulty measures for mQAPs: i) the covariance coefficients of the elements in two different flow matrices, and ii) the correlation between the cost functions of two objectives.

Dominance [7] is a measure of the amplitude of the variance for the flow matrix and distance matrix. Note that there are $m + 1$ dominance values: m flow dominance values and one distance dominance value. We denote the distance and flow dominances with $d_a = \frac{\sigma_a}{\mu_a} \%$ and $d_k = \frac{\sigma_{b^k}}{\mu_{b^k}} \%$, where μ_a and σ_a is the mean and the standard deviation for the matrix a . A matrix with low epistasis has the dominance close to the lower bound, 0. The dominance's upper bound is 100.

We propose to measure the amplitude of the sample covariance between two flow matrices, b^k and b^r . The *dominance* of the flow matrices b^k and b^r is defined as $d_{kr} = \frac{1}{\sqrt{\mu_{b^k} \cdot \mu_{b^r}}} \cdot \sqrt{\frac{\sum_{i,j=1}^n (b_{ij}^k - \mu_{b^k}) \cdot (b_{ij}^r - \mu_{b^r})}{n^2}} \%$.

Table 1. Analytical and empirical properties of 9 bi-objective QAP instances

type mQAPs	N	n	Dominance				Ruggedness		Asymptotic behavior			Empirical behavior			
			d_a	$d_{b,1}$	$d_{b,2}$	$d_{1,2}$	$\phi_{b,1}$	$\phi_{1,2}$	Best Know	Invers	Gap	% opt	mean	Gap	
Knowless ^r	25		60	64	63	18	96	92	646561					672236	3.8
	50		59	61	58	13	98	98	5264742					5333790	1.2
	75		60	59	60	13	99	98	12285680					12476382	1.5
Composite	25		94	99	95	97	95	95	183427	1381042	87	100	183427	87	
	50	1	92	97	93	94	98	97	737069	6010180	87	7	1965328	67	
	75		91	96	91	94	99	99	1923900	13417974	88	0	4766598	64	
	25		83	93	83	88	95	95	189465	1352778	86	100	189465	86	
	50	5	85	95	85	90	96	96	745782	5560362	87	12	1707298	69	
	75		83	92	83	87	98	98	3317244	12324804	73	0	4768865	61	
	20		83	97	90	93	98	98	244998	536258	54	100	244998	54	
	50	10	84	93	84	89	98	97	742401	4807446	85	1	1790843	63	
	80		82	92	83	87	98	98	3870974	14129836	73	0	5351231	62	

Ruggedness [10] is a normalisation of the *autocorrelation* coefficient for the cost function c^k when a (m)QAP is explored with local search. By definition, the autocorrelation coefficient for the k objective is $\epsilon_{c^k} = \frac{2 \cdot (\mathbb{E}[(c^k)^2] - \mu_{c^k}^2)}{\mathbb{E}[(c^k(\pi) - c^k(\pi'))^2]}$, where μ_{c^k} is the average of c^k and π and π' are *any* two permutations. The ruggedness coefficient for the k -th objective is $\phi_{b^k} = 100 - \frac{400}{n-2} \cdot (\epsilon_{b^k} - \frac{n}{4})$. A ruggedness coefficient close to 0 indicates a flat landscape, whereas a large ϕ^k , close to 100, indicates a steep landscape with lots of local optima.

We propose to measure the correlation between the cost functions of two objectives, c^k and c^r , $\epsilon_{kr} = \frac{\mathbb{E}[c^k(\pi) - \mu_{c^k}]}{\sigma_{c^k}} \cdot \frac{\mathbb{E}[c^r(\pi) - \mu_{c^r}]}{\sigma_{c^r}}$. The *ruggedness* of the objectives k and r is defined as $\phi_{kr} = 100 - \frac{400}{n-2} \cdot (\epsilon_{kr} - \frac{n}{4})$.

A difficult (m)QAP instance has both large dominance and ruggedness.

Asymptotic Behaviour of mQAPs calculates the difference between the optimum value (or a known feasible solution) and the value of the solution generated with the inverse permutation of that solution. Here, we consider the inverse of the optimum (or, if optimum is unknown, the best known solution) an approximation of the worst solution of an instance of composite mQAP. Thus, for mQAPs with the optimum solution \mathcal{I} , we assume that the reverse of the identity permutation, \mathcal{I}^{-1} , is an approximation of the worst solution for that instance. The gap is $\frac{\text{inverse_solution} - \text{best_known_solution}}{\text{inverse_solution}} * 100\%$, where the *best_known_solution* is the best solution returned by an algorithm and *inverse_solution* is the inverse solution for the best known so far.

Numerical Examples. Let consider the numerical example from Section 4. In Table 1 we compare the difficulty of several bi-objective QAP instances from [7] and the composite bi-objective QAPs. We consider the correlation between the flow matrices $\rho = 0.75$, and $N = \{25, 50, 75\}$. The asymptotic and empirical behaviour is shown only for the first objective. To compute the empirical behaviour of bQAPs we run iterated Pareto LS [8] for 50 times each run for 10^6 position swaps in a permutation. The composite bQAPs are most difficult tested instances because they have the largest dominance values, ruggedness coefficients and gaps. The small composite bQAPs, $N = 25$, have a lower ruggedness than the large composite bQAPs, $N = 75$. The dominance values and the gap decrease with the size increase of the component bQAPs because there

is less variance in the values of the component matrices. Note that the empirical gap is much smaller than the asymptotic gap. To conclude, the analytical (difficulty measures) and empirical properties of the composite bQAPs outperform the same properties of the uniformly randomly generated bQAPs.

6 Conclusion

We propose a multi-objective quadratic assignment instance generator that aggregates several small multi-objective QAP instances into a larger mQAP instance. Both the component mQAP instances and the cost of the elements outside these components have, by design, the identity permutation as the optimal solution. We give mild conditions under which the resulting composite mQAP instances have identity permutation as the optimum solution. We propose difficulty measures to compare the proposed composite mQAPs with other mQAPs from literature. We conclude that composite bQAP instances are more difficult than the uniform random bQAPs, and in addition, they have a known optimum solution.

References

1. Puglierin, F., Drugan, M.M., Wiering, M.: Bandit-inspired memetic algorithms for solving quadratic assignment problems. In: Proc. of CEC, pp. 2078–2085. IEEE (2013)
2. Drugan, M.M.: Cartesian product of scalarization functions for many-objective QAP instances with correlated flow matrices. In: Proc. of GECCO, pp. 527–534. ACM (2013)
3. Çela, E.: The Quadratic Assignment Problem: Theory and Algorithms. Kluwer Academic Publishers, Dordrecht (1998)
4. Drezner, Z., Hahn, P., Taillard, E.: Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. *Annals of Operations Research* 139(1), 65–94 (2005)
5. Drugan, M.: Generating QAP instances with known optimum solution and additively decomposable cost function. *Journal of Combinatorial Optimization* (2014)
6. Palubeckis, G.: An algorithm for construction of test cases for the quadratic assignment problem. *Informatica, Lith. Acad. Sci.* 11(3), 281–296 (2000)
7. Knowles, J.D., Corne, D.W.: Instance generators and test suites for the multiobjective quadratic assignment problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 295–310. Springer, Heidelberg (2003)
8. Drugan, M.M., Thierens, D.: Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies. *J. Heuristics* 18(5), 727–766 (2012)
9. Wayne, A.: Inequalities and inversions of order. *Scripta Mathematica* 12(2), 164–169 (1946)
10. Angel, E., Zissimopoulos, V.: On the hardness of the quadratic assignment problem with metaheuristics. *J. Heuristics* 8(4), 399–414 (2002)

Optimized Approximation Sets for Low-Dimensional Benchmark Pareto Fronts

Tobias Glasmachers

Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany
tobias.glasmlachers@ini.rub.de

Abstract. The problem of finding sets of bounded cardinality maximizing dominated hypervolume is considered for explicitly parameterized Pareto fronts of multi-objective optimization problems. A parameterization of the Pareto front it often known (by construction) for synthetic benchmark functions. For the widely used ZDT and DTLZ families of benchmarks close-to-optimal sets have been obtained only for two objectives, although the three-objective variants of the DTLZ problems are frequently applied. Knowledge of the dominated hypervolume theoretically achievable with an approximation set of fixed cardinality facilitates judgment of (differences in) optimization results and the choice of stopping criteria, two important design decisions of empirical studies. The present paper aims to close this gap. An efficient optimization strategy is presented for two and three objectives. Optimized sets are provided for standard benchmarks.

1 Introduction

Empirical benchmark studies play a major role for performance comparisons of nature inspired (optimization) algorithms. For many benchmark problems in widespread use the optimum is known analytically. This allows to compare algorithms not only relative to each other but also in relation to the actual optimum. This is a prerequisite, e.g., for the empirical investigation of convergence rates. It is practically useful for the design of meaningful stopping criteria in benchmark studies comparing the runtime of different (black-box) optimization algorithms for reaching a predefined solution accuracy.

The situation in multi-objective optimization differs in various respects from the single-objective case. The optimum is a set—the Pareto front—which can be of uncountably infinite cardinality. In practice optimal subsets of a priori bounded cardinality are of primary interest. There are multiple performance indicators in common use, and the optimal set of course depends on the indicator. In recent years the hypervolume indicator has advanced to the most widely applied performance measure at least for up to three or four objectives. Hence this study is focused on maximization of dominated hypervolume.

For the commonly used ZDT and DTLZ benchmark suites [13,7] sets with close-to-optimal hypervolume coverage are known only for the simplest case of

two objectives [1]. However, the DTLZ benchmarks are scalable to any number of objectives, with a default of three objective functions [7].

The present study aims to close this gap. For this purpose an efficient gradient-based hypervolume maximization algorithm based on an explicit parameterization of the Pareto front is proposed. With this algorithm we compute optimized fixed cardinality Pareto front approximations for bi-objective and tri-objective benchmark problems.

2 Multi-objective Optimization

Consider a search space X and a set of m scalar objective functions $f_1, \dots, f_m : X \rightarrow \mathbb{R}$, each of which (w.l.o.g.) is to be minimized. Instead of aggregating the various goals encoded by the different functions into a single objective (e.g. by means of a weighted combination), the goal of multi-objective optimization (MOO) is to obtain the set of Pareto optimal (or non-dominated) points, which is the set of Pareto optimal compromises. This set is often huge or even infinite and we aim for a representative approximation set of a-priori bounded cardinality.

2.1 Dominance Order and Dominated Hypervolume

The objectives are collected in the vector-valued objective function $f : X \rightarrow \mathbb{R}^m$, $f(x) = (f_1(x), \dots, f_m(x))$. Let $Y = \{f(x) \mid x \in X\} = f(X) \subset \mathbb{R}^m$ denote the image of the objective function (also called the attainable objective space). For values $y, y' \in \mathbb{R}^m$ we define the Pareto dominance relation

$$\begin{aligned} y \preceq y' &\Leftrightarrow y_k \leq y'_k \text{ for all } k \in \{1, \dots, m\} \text{ ,} \\ y \prec y' &\Leftrightarrow y \preceq y' \text{ and } y \neq y' \text{ .} \end{aligned}$$

This relation defines a partial order on Y , incomparable values y, y' fulfilling $y \not\preceq y'$ and $y' \not\preceq y$ remain. The relation is pulled back to the search space X by the definition $x \preceq x'$ iff $f(x) \preceq f(x')$.

The Pareto front is defined as the set of values that are optimal w.r.t. Pareto dominance, i.e., the set of non-dominated values

$$Y^* = \left\{ y \in Y \mid \nexists y' \in Y : y' \prec y \right\}$$

and the Pareto set is $X^* = f^{-1}(Y^*)$.

For generic objectives f_k without simultaneous critical points the Pareto front is a manifold of dimension $m - 1$. As such its cardinality is uncountably infinite. We are often interested in picking a “representative subset” or approximation set of fixed cardinality n . The approximation quality of a set $S = \{y_1, \dots, y_n\}$ can be judged with different set quality indicators of which the dominated hypervolume

$$H_r(S) = \Lambda\left(\{y' \in \mathbb{R}^m \mid \exists y \in S \text{ s.t. } y \preceq y' \preceq r\}\right)$$

(where Λ denotes the Lebesgue measure on \mathbb{R}^m) is distinguished (up to weighting of the Lebesgue measure) for its property of being compliant with Pareto dominance [14,12]. The hypervolume indicator depends on a reference point $r \in \mathbb{R}^m$ that needs to be set by the user. It defines an objective-wise cut-off for the quality assessment.

One standard formalization of the goal of MOO is to produce a set $\{x_1, \dots, x_n\}$ of n points so that the corresponding values $y_k = f(x_k)$ maximize the hypervolume indicator for given f , r , and n . It is easy to see that the elements of the set S^* maximizing the hypervolume $H_r(S)$ are Pareto optimal, i.e., $S^* = \{y_1, \dots, y_n\} \subset Y^*$.

The optimal set S^* as well as the dominated hypervolume $H_r(S^*)$ depend on X and f only via Y^* . Since we are interested only in S^* and $H_r(S^*)$ we simplify the problem statement in the following by assuming that the set Y^* is known. This means that we will ignore a large part of the complexity of the underlying MOO problem related to the black-box setting and the potentially involved form of f . For practical purposes we may assume that a (surjective, sometimes bijective) parameterization $\varphi : U \rightarrow Y^*$, $U \subset \mathbb{R}^{m-1}$, is available. This parameterization then replaces the (often much harder to optimize) objective function.

2.2 Benchmark Problems and Existing Results

In this study we focus on well-established benchmark suits of problems with continuous variables. The so-called ZDT functions [13] ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 (ZDT5 is a discrete problem) are scalable to any search space dimension $X = \mathbb{R}^d$ but come with only two objectives. The conceptually similar DTLZ functions [7] DTLZ1, DTLZ2, DTLZ3, and DTLZ4 (the other members of the family are rarely used) improve on this situation by being scalable to many objectives, with the “recommended default” of three objectives.

For the ZDT and DTLZ families of problems the sets Y^* are known analytically. For $m = 2$ near optimal sets S of cardinalities 2, 3, 4, 5, 10, 20, 50, 100, 1000 for the reference point (11, 11) have been obtained, see [1]. The one-dimensional fronts are visualized in figure 1 (a) to (f).

For $m = 3$ the DTLZ fronts become two-dimensional. They are depicted in figure 1 (g) and (h). Fixed size sets with maximal hypervolume coverage are not known. We provide such near optimal sets in section 5 and in the supplementary material.

The problem of optimizing $H_r(S)$ has been investigated theoretically, but analysis is mostly restricted to two objectives [3,4,11]. Results include conditions for the inclusion of extremal points in the optimal solution set, monotonicity of $H_r(S^*)$ in $n = |S^*|$, optimal sets on linear fronts, and asymptotically optimal distributions on smooth fronts in the limit $n \rightarrow \infty$. Basic results have been obtained for $m = 3$ objectives [2]; but even asymptotically optimal distributions based on local shape (derivative) features are unknown.

Finally, the most relevant precursors for obtaining optimal sets of fixed cardinality n in practice are algorithms for the fast computation of the hypervolume and its derivatives [10,9,8].

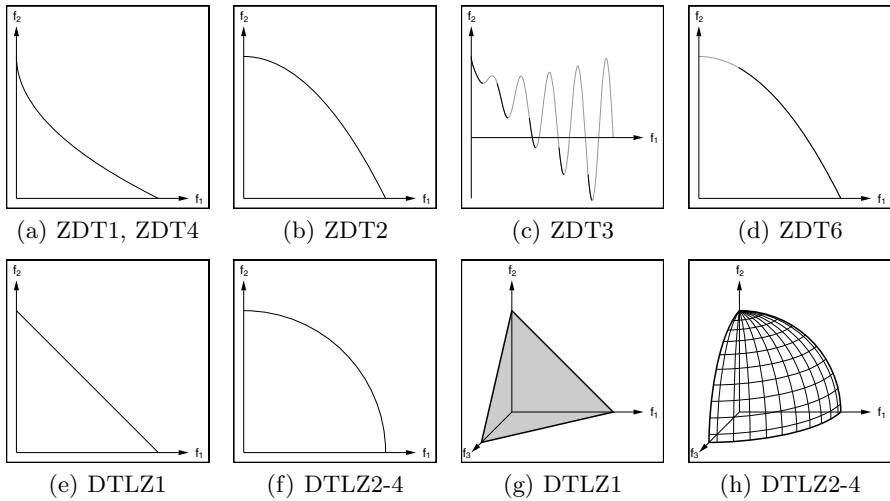


Fig. 1. Pareto fronts of the ZDT and DTLZ problems for $m = 2$ and $m = 3$

3 Hypervolume Calculation

The computation of the dominated hypervolume $H_r(S)$ of n values $S = \{y_1, \dots, y_n\} \subset Y^*$ is known to be NP-hard, and the time complexity of the best known algorithms for this problem is exponential in m [5,6]. However, for few objectives ($m \leq 3$) it can be carried out in $\mathcal{O}(n \log(n))$ operations [10,9,8].

In the present paper we consider only subsets $S \subset Y^*$. Thus any pair of different points $y_i, y_j \in S$ is strictly incomparable (i.e., it holds $y_i \not\leq y_j$ and $y_j \not\leq y_i$).

3.1 Gradient of Dominated Hypervolume

It is easy to see that the dominated hypervolume $H_r(S)$ for $S = \{y_1, \dots, y_n\}$ considered as a function of y_k is nearly everywhere differentiable. Efficient algorithms for the computation of the hypervolume and its derivatives for $m \leq 4$ objectives have been proposed in [8]. It is sufficient for the purpose of practical optimization to consider the vector of derivatives $\frac{\partial H_r(S)}{\partial y_k}$, $k \in \{1, \dots, n\}$, and to ignore issues of the argument S being an *unordered* set of values (refer to [8] for a more careful derivation). With $y_k = \varphi(u_k)$ the chain rule gives rise to $\frac{\partial H_r(S)}{\partial u_k} = \frac{\partial H_r(S)}{\partial y_k} \frac{\partial y_k}{\partial u_k}$ with $\frac{\partial y_k}{\partial u_k} = \varphi'(u_k)$. Note that in our setting the gradient does not aid in finding the Pareto front, since the front is already implicitly encoded in the parameterization φ . Instead it indicates how to improve the spread (distribution for maximal hypervolume) of the set S (e.g., by means a gradient ascent step).

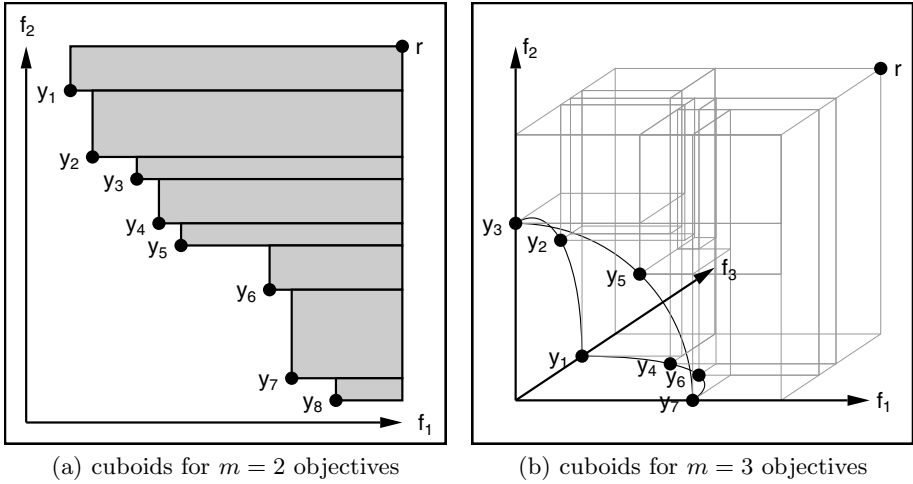


Fig. 2. Decomposition of the dominated hypervolume into disjoint cuboids

3.2 Decomposition into Cuboids

In this section we emphasize the possibility to represent the dominated hypervolume explicitly as a disjoint¹ union of simple volumes, in this case m -dimensional axis-aligned cuboids (rectangles in two dimensions, cuboids in three dimensions). We restrict the following consideration to $m = 2$ and $m = 3$. A crucial observation is that the cuboids are not only aligned to the coordinate axes but also start and end at objective values that appear in the set $S^+ = S \cup \{r\}$. Each rectangle can thus be represented as a $2m$ -tuple of values² $(y_1^-, y_1^+, \dots, y_m^-, y_m^+)$ representing the cuboid $[(y_1^-)_1, (y_1^+)_1] \times \dots \times [(y_m^-)_m, (y_m^+)_m] \subset \mathbb{R}^m$.

For $m = 2$ the set S is sorted by one objective, which results in the reverse order in the other objective. Then the hypervolume is computed by splitting the dominated set into disjoint rectangles and summing their areas (see figure 2 (a)). Thus the decomposition into exactly n rectangles is a cheap by-product of the hypervolume computation: sorting the points requires $\mathcal{O}(n \log(n))$ operations, while there are only n rectangles.

For $m = 3$ we adopt the sweep-based algorithm from [10]. Instead of computing the hypervolume on the fly the modified algorithm stores and reports a list of n to $2n - 1$ and hence $\Theta(n)$ cuboids. So again the collection of the cuboids is a rather cheap by-product of the $\mathcal{O}(n \log(n))$ hypervolume computation. The result is illustrated in figure 2 (b).

The decomposition into cuboids is not more costly than the computation of the hypervolume itself, and for $m \leq 3$ the number of cuboids is only linear in n .

¹ For simplicity of presentation the cuboids' boundaries may overlap. However, the corresponding open cuboids (the topological interiors) are disjoint. This does not impact the hypervolume computation.

² For efficiency reasons, indices or pointers may be used in actual implementations.

This proceeding has the advantage that subsequent tasks such as the computation of the hypervolume or its partial derivatives (possibly higher derivatives) [8], dependence of hypervolume contributions on other points [9], etc. do not need to be incorporated into the bookkeeping-heavy sweeping algorithm. Instead they can be realized in a unified scheme, namely by first invoking the cuboid decomposition algorithm and a subsequent loop over the list of cuboids.

4 Hypervolume Optimization

4.1 Gradient-Based Optimization

In the following we assume an algorithm that represents the dominated hypervolume as a disjoint union of m -dimensional axis aligned cuboids as discussed in section 3. This allows for the trivial computation of $H_r(S)$ as the sum of the elementary volumes.

The coordinate wise lower and upper bounds of the cuboids are given by coordinates of points from the set $S^+ = S \cup \{r\}$. This representation allows for trivial differentiation of $H_r(S)$ w.r.t. points in S , yielding n derivatives $\frac{\partial H_r(S)}{\partial y_k}$ (see also section 3.1). Thus, starting from any initial configuration S can be iteratively refined with a gradient-based optimization procedure.

We propose simple gradient ascent steps $u_k \leftarrow u_k + \eta \cdot \nabla_{u_k} H_r(S)$ with learning rate $\eta > 0$. If a step happens to decrease the hypervolume then backtracking is applied: the previous set S is restored and the learning rate η is halved. Otherwise the learning rate is optimistically increased (here by a factor of 1.05). This heuristic is analog to success-based step-size control in elitist evolution strategies.

4.2 Dealing with Multi-modality

Gradient ascent is an efficient technique for the localization of a *local* maximum. It turns out that for $m > 2$ multi-modality of the hypervolume indicator is a practically relevant issue. We address this problem with two simple yet effective techniques: proper initialization and a multi-start strategy.

Uniform initialization of the parameters $\{u_1, \dots, u_n\} \subset U$ may lead to a highly distorted distribution of the actual values $\{\varphi(u_1), \dots, \varphi(u_n)\} \subset Y^*$. Furthermore, at least in the limit $n \rightarrow \infty$ the density of values should depend on the slope of the front. For $m = 2$ objectives this was derived in [3]. Here we present a heuristic (inexact yet practical) extension of these ideas to $m = 3$ objectives. Importantly, we do not claim to solve the problem of asymptotically optimal distributions but rather aim for a procedure generating suitable initial solutions for gradient-based optimization.

For simplicity let us assume that the hypervolume contribution of a point $y = \varphi(u)$ consists of the volume of a cuboid with side lengths b_1, b_2, b_3 and volume $V = b_1 \cdot b_2 \cdot b_3$. Furthermore assume that the point is not close to the boundary of the front and that the front surface is regular at y . This implies that

locally the front can be approximated by a hyperplane (here a plane), spanned by the derivative vectors $\frac{\partial\varphi}{\partial u_1}$ and $\frac{\partial\varphi}{\partial u_2}$. Its orientation is characterized by the normal vector $w(u) = \frac{\partial\varphi}{\partial u_1} \times \frac{\partial\varphi}{\partial u_2}$, which can be obtained as the cross product of the tangent vectors. The norm $\|w\|$ measures the (local) volume growth of φ .

For large n the normal vector determines the local distribution of values y which—in the optimum—locally have equal hypervolume contributions $V \approx \text{const}$. At the same time it should hold $w_1 b_1 \approx w_2 b_2 \approx w_3 b_3$ for the shape of the cuboid. Putting these together we obtain that b_i is proportional to $\sqrt[3]{w_1 w_2 w_3 / w_i}$. The “area of the front” covered by the cuboid can be approximated by the area of the triangle spanned by the cuboid vertices $y + (b_1, 0, 0)$, $y + (0, b_2, 0)$, and $y + (0, 0, b_3)$. It is computed as $A = \frac{1}{2} \cdot \sqrt{b_1^2 b_2^2 + b_1^2 b_3^2 + b_2^2 b_3^2}$. Under the above considerations the optimal density of parameters u is proportional to $\|w\|/A$.

We sample an initial set of size n from the above density by means of rejection sampling. For this purpose $\kappa = 100$ random points are drawn from the uniform distribution on $u \in U$. The maximal value of $\|w\|/A$ over these points, multiplied by a safety margin of two, is kept as a tentative upper bound B on the unnormalized density. Then parameters u are sampled uniformly and rejected with probability $\min\{1, 1 - \|w\|/(A \cdot B)\}$ until n samples are accepted.

Our multi-start procedure generates N independent initial sets of size n as described above. Each initial set serves as a starting point for the gradient-based optimization algorithm.

4.3 Implementation

We provide an efficient C++ implementation of the above described hypervolume maximization algorithm with rejection sampling initialization and restart strategy. The program can be downloaded from <http://www.ini.rub.de/PEOPLE/glasmtbl/code/opt-hv/>.

5 Optimized Sets for the ZDT and DTLZ Problems

In this section we present the close to optimal sets obtained by our optimization algorithm for the ZDT and DTLZ problems. In analogy to [1] we have conducted all experiments for cardinalities $n \in \{2, 3, 4, 5, 10, 20, 50, 100, 1000\}$.

5.1 The Bi-objective Case

The ZDT and DTLZ problems have been optimized with specifically tailored algorithms. The results are found on the website [1], which is a valuable resource when experimenting with these benchmark problems. We stick to the original reference point $r = (11, 11)$. This test aims to validate our optimization algorithm.

We have run the gradient-based optimization procedure $N = 100$ times with random initial configurations. The results are presented concisely in table 1. Standard deviations across repetitions are extremely small (usually below 10^{-10}). The global optimum is obtained in each single run. Most of our results reproduce

Table 1. Maximal dominated hypervolume covered by sets of cardinalities $n \in \{2, 3, 4, 5, 10, 20, 50, 100, 1000\}$ for bi-objective problems with reference point $(11, 11)$

n	ZDT1,4	ZDT2	ZDT3	ZDT6	DTLZ1	DTLZ2-4
2	120.0248764	120.0000000	128.0147714	117.2489467	120.7500000	120.0000000
3	120.3877279	120.1481481	128.4523400	117.3723140	120.8125000	120.0857864
4	120.4915975	120.2041588	128.5997409	117.4178988	120.8333333	120.1215851
5	120.5397291	120.2339071	128.6671568	117.4417417	120.8437500	120.1415358
10	120.6137609	120.2868199	128.7459431	117.4832459	120.8611111	120.1789660
20	120.6423963	120.3106986	128.7632012	117.5014399	120.8684211	120.1968576
50	120.6574465	120.3243978	128.7707848	117.5116580	120.8724490	120.2074851
100	120.6621372	120.3288807	128.7739496	117.5149559	120.8737374	120.2110337
1000	120.6662212	120.3328889	128.7774084	117.5178796	120.8748749	120.2142433

the optimized fronts obtained in [1]. For large values of n we observe slight improvements. For example, for problem DTLZ2 with $n = 100$ our gradient-based procedure obtains a dominated hypervolume of 120.2110337 instead of the previously reported value of 120.210644. The improvement in itself may seem minor, however, for $n = 100$ and $n = 1000$ our optimization procedure improves on most of the existing numbers.

The ZDT3 problem is an exception. Here we observe improved values for small n . This is because the left extreme point should not be fixed for the optimization, see also Theorem 2 in [3]. On the other hand our results for large n are significantly worse than those reported at [1] since gradient-ascent cannot deal well with the disconnected front of the ZDT3 problem and the resulting discontinuous parameterization φ .

5.2 The Tri-objective Case

A major motivation for the present work is to obtain optimized fronts for the DTLZ problems in their standard form, which is with three objectives. Results of our gradient-based optimizer are presented in table 2. The optimized sets are available for download at <http://www.ini.rub.de/PEOPLE/glasmtb1/code/opt-hv/> in csv format, and as eps and png figures.

The variance in the results is significantly higher than in the bi-objective case. This is because of the multi-modality of the problem. Hence we have increased the number of runs to $N = 10,000$. Running the procedure with even more repetitions will most probably give slightly higher hypervolumes. However, most reasonable optimization procedures may get stuck in local optima. Therefore not only the global optimum is of interest but also the distribution of local optima. The descriptive statistics in table 2 provide such data. This allows to judge the performance of algorithm on an absolute scale w.r.t. a reference distribution, e.g., by measuring how often a certain quantile of the empirical distribution of local optima is reached.

Table 2. Characteristics (mean, standard deviation, quantiles, and maximum) of the empirical distributions of dominated hypervolume for the DTLZ1 front (upper half) and the DTLZ2-4 front (lower half) with $m = 3$ objectives, reference point $r = (2, 2, 2)$, and cardinalities $n \in \{2, 3, 4, 5, 10, 20, 50, 100, 1000\}$

front	n	mean	stddev	25%	50%	75%	max	
DTLZ1	2	7.5281031	0.0094041	7.5312500	7.5312500	7.5312500	7.5312500	
	3	7.8750000	0.1102255	7.6445649	7.8750000	7.8750000	7.8750000	
	4	7.8946157	0.0526134	7.9062500	7.9062500	7.9062500	7.9120370	
	5	7.9222298	0.0212193	7.9238281	7.9242346	7.9259728	7.9260397	
	10	7.9532612	0.0005410	7.9529850	7.9532053	7.9537283	7.9539787	
	20	7.9644361	0.0001552	7.9643638	7.9644671	7.9645441	7.9647401	
	50	7.9712554	0.0000490	7.9712280	7.9712615	7.9712901	7.9713876	
	100	7.9739706	0.0000267	7.9739557	7.9739739	7.9739892	7.9740466	
	1000	7.9776989	0.0000039	7.9776965	7.9776992	7.9777016	7.9777110	
DTLZ2	2	6.0000000	0.0000000	6.0000000	6.0000000	6.0000000	6.0000000	
	3	6.8272558	0.3365748	7.0000000	7.0000000	7.0000000	7.0000000	
	4	7.0694591	0.1090694	7.0857864	7.0857864	7.0857864	7.0857864	
	5	7.1467484	0.0206817	7.1493061	7.1493061	7.1493061	7.1493061	
	DTLZ3	10	7.2809948	0.0049230	7.2780682	7.2795647	7.2860090	7.2874732
		20	7.3485703	0.0022931	7.3472972	7.3488734	7.3501758	7.3545152
	DTLZ4	50	7.3994118	0.0010188	7.3987853	7.3995002	7.4001307	7.4022754
		100	7.4228644	0.0006244	7.4224787	7.4229145	7.4232955	7.4246456
		1000	7.4597704	0.0001156	7.4596963	7.4597782	7.4598501	7.4601203

6 Conclusion

We have presented an efficient algorithm for the maximization of dominated hypervolume of sets of fixed cardinality when a parametric form of the Pareto front is known. Such sets are of practical relevance when comparing multi-objective optimizers on benchmark problems. While existing studies have been restricted to relative comparisons we are now in the position to relate differences to an absolute scale given by the best known hypervolume and by the empirical distribution of local optima as identified by our multi-start procedure. This also allows to report the performance of a single (e.g., novel) algorithm on an absolute scale rather than relative to (arbitrarily chosen) competitors. Our gradient-based procedure is computationally efficient. This algorithm has been integrated into a standalone software with easy-to-use command line interface.

References

1. ZDT and DTLZ test problems, <http://people.ee.ethz.ch/~sop/download/supplementary/testproblems/> (accessed: March 17, 2014)
2. Auger, A., Bader, J., Brockhoff, D.: Theoretically Investigating Optimal μ -Distributions for the Hypervolume Indicator: First Results for Three Objectives. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 586–596. Springer, Heidelberg (2010)

3. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the Hypervolume Indicator: Optimal μ -Distributions and the Choice of the Reference Point. In: Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, pp. 87–102. ACM (2009)
4. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Hypervolume-based Multiobjective Optimization: Theoretical Foundations and Practical Implications. *Theoretical Computer Science* 425, 75–103 (2012)
5. Bringmann, K.: Klee’s measure problem on fat boxes in time $\mathcal{O}(n(d+2)/3)$. In: Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry, pp. 222–229. ACM (2010)
6. Bringmann, K., Friedrich, T.: Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. *Theoretical Computer Science* 425, 104–116 (2012)
7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Congress on Evolutionary Computation, CEC 2002, pp. 825–830. IEEE Press (2002)
8. Emmerich, M., Deutz, A.: Time complexity and zeros of the hypervolume indicator gradient field. In: Schuetze, O., Coello Coello, C.A., Tantar, A.-A., Tantar, E., Bouvry, P., Del Moral, P., Legrand, P. (eds.) *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III*. SCI, vol. 500, pp. 169–193. Springer, Heidelberg (2014)
9. Emmerich, M.T.M., Fonseca, C.M.: Computing hypervolume contributions in low dimensions: Asymptotically optimal algorithm and complexity results. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) *EMO 2011*. LNCS, vol. 6576, pp. 121–135. Springer, Heidelberg (2011)
10. Fonseca, C.M., Paquete, L., Lopez-Ibanez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: *IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 1157–1163 (2006)
11. Friedrich, T., Neumann, F., Thyssen, C.: Multiplicative Approximations, Optimal Hypervolume Distributions, and the Choice of the Reference Point. Technical Report arXiv:1309.3816, arXiv.org (2013)
12. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) *EMO 2007*. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)
13. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation* 8(2), 173–195 (2000)
14. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7(2), 117–132 (2003)

Start Small, Grow Big?

Saving Multi-objective Function Evaluations

Tobias Glasmachers¹, Boris Naujoks², and Günter Rudolph³

¹ Ruhr-Universität Bochum, Germany
tobias.glasachers@ini.rub.de

² Cologne University of Applied Sciences, Germany
boris.naujoks@fh-koeln.de

³ Technische Universität Dortmund, Germany
guenter.rudolph@tu-dortmund.de

Abstract. The influence of non-constant population sizes in evolutionary multi-objective optimization algorithms is investigated. In contrast to evolutionary single-objective optimization algorithms an increasing population size is considered beneficial when approaching the Pareto-front. Firstly, different deterministic schedules are tested, featuring different parameters like the initial population size. Secondly, a simple adaptation method is proposed. Considering all results, an increasing population size during an evolutionary multi-objective optimization algorithm run saves fitness function evaluations compared to a fixed population size. In particular, the results obtained with the adaptive method are most promising.

1 Introduction

The size of the population is an external parameter in evolutionary algorithms (EA, [6,7]). Chosen once, it is expected to stay constant for the whole optimization run. However, the right choice of the population size has an enormous effect on the outcome of the EA run. Results vary from very good to very poor only with respect to a proper setting. Choosing a population too small may prevent the localization of optimal solutions, whereas choosing a population too large wastes considerable resources, in particular if fitness function evaluations are computationally expensive. Thus, a dynamic population size might help in saving many function evaluations without any loss in solution quality.

A typical single-objective EA run can be split into two phases. During the first phase the aim of the algorithm is to identify the basin of the globally best solution. To this end, a larger population size seems to be adequate. Having identified this basin or at least a good candidate for it, the goal of the algorithm shifts to identifying the best solution within this basin. Here, a smaller population size is sufficient. This possibly scales down to the $(1 + 1)$ selection scheme.

The situation changes completely if more than one objective is considered. The optimization run of an evolutionary multi-objective optimization algorithm

(EMOA, [3,4]) can be split into two phases as well. However, the population sizes are expected to be adapted best in a different way. Often, only a rather limited population size is sufficient to approach a Pareto-front. Once the front is reached, it needs to be explored and covered well, and, consequently, larger populations are expected to perform better at this task.

Another difference is the role of the population in single and multi-objective EA. In a single-objective EA, the population allows to sample the fitness landscape sufficiently well, and to maintain a set of diverse solutions. The performance of the population is measured by the fitness of the best individual. In a multi-objective EA, the population represents the algorithm's current approximation to the Pareto-front. All non-dominated individuals contribute to it's performance. Usually, there is a pre-defined upper limit on the allowable size of the result set, which is often considered the canonical value of the population size parameter.

As a consequence, the simple rule for adapting the population size of an EMOA could be: *start small, grow big*. Big refers here to the limit of the result set size, while we'd like to start with a much smaller size. In this study, we aim at proving that increasing the population size like this saves fitness function evaluations compared to a fixed population size.

Efforts for realizing a dynamic population size in evolutionary algorithms have been driven mainly by the desires to eliminate an external parameter and to improve performance. Only few attempts have been made to extend automatic population size control to *multi-objective* optimization. An early example (1977) of a dynamic population size in an EMOA was given by Peschel and Riedel [12], where the new population was formed by the non-dominated individuals from the union of parents and offspring. The PR1 algorithm [13] as well as the SEMO algorithm [10] work similarly. Tan et al. [14] as well as Lu and Yen [11] impose a cellular structure on the Pareto-front. This approach required prior knowledge of the front, which is only practical for synthetic benchmark problems. In [8] the population size develops as a function of a pre-defined time-dependent schedule (deterministic component) and the number of non-dominated individuals (adaptive component). It is maybe closest in spirit to the present study.

This study starts summarizing multi-criteria optimization basics before the schedules are defined in section 3. Section 4 presents experiments and results before we conclude our findings and provide a short outlook.

2 Multi-Criteria Optimization

Considering only one objective in applied optimization is a simplification that does not mirror the complexity of the underlying application in most (or almost all) cases. Often multiple objectives $f_1, \dots, f_n : X \rightarrow \mathbb{R}$ need to be considered. Here we focus on two objectives. The most common way to deal with multiple objectives appears to be aggregation, e.g., to a weighted sum $f(x) = \sum_i w_i f_i(x)$. In contrast, multi-objective or multi-criteria optimization (MCO) offers a different way to handle multiple objectives in a more unbiased, maybe more

effective way. For this approach, we have to consider the vector-valued objective function $f : X \rightarrow \mathbb{R}^n$, $f(x) = (f_1(x), \dots, f_n(x))$.

In MCO, an important concept is Pareto dominance, i.e., an objective value $y \in \mathbb{R}^n$ dominates another value $y' \in \mathbb{R}^n$ iff y is better in at least one dimension of the objective space and not worse in all the others. More formally and considering minimization, this reads

$$y \prec y' \quad \text{iff} \quad \forall i : y_i \leq y'_i \quad \wedge \quad \exists j : y_j < y'_j .$$

If an objective value y is not dominated by any other value in the image $A = f(X)$ of the objective function (or generated by the algorithm), it is said to be *non-dominated*, i.e., $\forall y' \in A : y' \not\prec y$. This concept allows for ranking of sets in the multi-dimensional objective space. MCO algorithms aim for the optimal set $A^* = \{y \in A \mid \nexists y' \in A : y' \prec y\}$. It has the property that every two points y and y' from A^* are mutually non-dominated, i.e. $y' \not\prec y \wedge y \not\prec y'$. This set is called the *Pareto-front*. The dominance relation is pulled back to the decision space X via the objective function by defining $x \prec x'$ iff $f(x) \prec f(x')$ for $x, x' \in X$. The resulting set $f^{-1}(A^*) \subset X$ of optimal solutions is called the *Pareto-set*.

In addition to the number of objectives, there is a structural change in the step from one to multiple objectives. The strict order of objective values in the single-criterion objective space turns into a partial order (induced by Pareto dominance) in the multi-criteria objective space. This structural change implies that besides Pareto dominance a secondary quality indicator is required for ranking and thus for rank-based selection in EA.

In recent years, the hypervolume [15,17] set indicator turned from a frequently used quality indicator to a well-established selection operator for EMOA. The hypervolume of a set Y is defined as the n -dimensional volume of the space spanned by the set and a reference point y_{ref} that needs to be defined by the user:

$$\Lambda \left(\bigcup_{y \in Y} \{y' \in \mathbb{R}^n \mid y \prec y' \prec y_{\text{ref}}\} \right)$$

with Λ being the n -dimensional Lebesgue measure of the given set. The hypervolume of a set $P \subset X$ of solutions (e.g., a population) is the hypervolume of the corresponding values $\{f(x) \mid x \in P\}$.

Maximization of the hypervolume covered by the population implicitly covers the traditional goals of convergence of the solution set to the optimal front as well as good solution spread. Most prominent instances of hypervolume based selection MCO algorithms are SMS-EMOA [2], Hyp-E [1], as well as MO-CMA-ES [9].

The $(\mu + 1)$ selection mechanism in SMS-EMOA provides an elegant way to enlarge and diminish the population size online. The population can be enlarged by skipping the selection step, and it can be reduced by skipping the offspring generation step. Therefore, this algorithm is used for the present study¹.

¹ The software is available on request by email to the first author.

3 Schedules

A population size schedule is simply a rule defining which population size to use at which time. Such a rule may be a fixed function of the generation counter or an adaptive decision rule based on online indicators. We investigate both possibilities.

3.1 Fixed Schedules

We start with the definition of a family of fixed, parametric schedules. The (increasing) population size is a function of time, measured by the number of fitness evaluations (FE), and normalized in relation to a budget of FE_{budget} fitness evaluations. In an application, this budget may be set to the affordable number of fitness evaluations. We think of it as a conservative estimate. We want to note clearly that the budget is a highly problem specific parameter. It is hard to guess a sound value without prior experimentation. Therefore, the requirement of providing an optimal budget parameter may not be realistic in practice. Here, this proceeding allows us to define comparable schedules for very different problems. As a practical solution, we also propose an adaptive scheduling strategy below.

The final population size S_{full} is the desired size of the Pareto-front approximation. In this study, it is fixed to $S_{\text{full}} = 100$. Besides these constants, each schedule is defined by four parameters $\alpha, \beta, \gamma, \delta \in [0, 1]$ as follows. The initial population size is set to $S_{\text{start}} = \lfloor \alpha \cdot S_{\text{full}} \rfloor$. The time FE_{full} by which the growing population size reaches S_{full} is represented as a fraction of the budget: $FE_{\text{full}} = \lfloor \beta \cdot FE_{\text{budget}} \rfloor$. In between we interpolate linearly. Thus, these parameters define linearly growing schedules with a cut-off at S_{full} . This class of schedules is further enriched by an intermediate point $(FE_{\text{inter}}, S_{\text{inter}})$, defined by $FE_{\text{inter}} = \lfloor \gamma \cdot FE_{\text{full}} \rfloor$ and $S_{\text{inter}} = \lfloor (1 - \delta) \cdot S_{\text{start}} + \delta \cdot S_{\text{full}} \rfloor$. This construction is illustrated in figure 1.

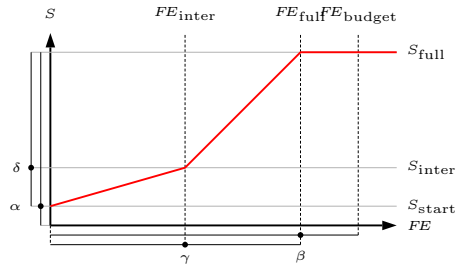


Fig. 1. Illustration of the fixed schedules and their parameters as functions of the population size S over the number of fitness evaluations FE . Refer to the text for details.

3.2 Adaptive Schedules

An alternative to a fixed schedule as a function of FE is an adaptive rule. We have found the following extremely simple rule to be effective: maintain a fading record of success probabilities, and increase the population size by a factor as soon as the success rate drops below a threshold. This rule is inspired by Rechenberg’s famous $1/5$ rule for the adaptation of the step size (mutation strength)

in evolution strategies. Here, a very similar rule is used successfully in a completely different context.

The algorithm works as follows. The low-pass filtered success rate is initialized to a value of $R \leftarrow 1/2$. In each generation it is updated according to $R \leftarrow (1 - \eta) \cdot R + \eta \cdot \mathbf{1}_{\text{suc}}$ where the success indicator $\mathbf{1}_{\text{suc}}$ is one if the offspring generated in the current iteration survives the selection phase and zero otherwise. A learning rate of $\eta = 0.01$ results in sufficiently stable behavior. Once the success rate indicator R drops below the threshold of $1/5$, the population size is increased:

$$S \leftarrow \min \left\{ \lceil c \cdot S \rceil, S_{\text{final}} \right\}, \quad R \leftarrow 1/4$$

At the same time, the success rate R is reset away from $1/5$ in order to avoid multiple population size increases due to random effects. We set the increase factor to $c = 3/2$.

The intuition behind this scheme is that initially the success rate is high, since all individuals are selected with the same probability. As the population approaches the front, successes become harder to sample as the success probability slowly approaches zero. At this stage, progress can be made only by spreading out the population over the front, which requires an increase of the population size.

This adaptive strategy has a number of parameters such as the initial value of the success rate estimate, the success rate threshold of $1/5$, the learning rate of 0.01 and the increase factor of $3/2$. We did not tune these parameters. We have run a few trials with other parameter settings and we did not find the algorithm to be very sensitive to the exact values. However, the threshold should be kept around $1/5$ for the procedure to work well.

The only remaining critical parameter is the initial population size S_{start} . Analog to the fixed schedules defined above, we express this parameter by means of $\alpha \in [0, 1]$ as $S_{\text{start}} = \lfloor \alpha \cdot S_{\text{full}} \rfloor$.

4 Experimental Evaluation

The goal of our experimental evaluation is two-fold. First of all, we aim for an overview of whether and how many fitness evaluations can be saved with a non-uniform population size schedule. To this end, we test a large number of deterministic schedules against the baseline method, which is to run the EMOA with the full target population size. Second, and maybe more importantly, we investigate the performance of our adaptive population size control method. The primary performance comparison is with the uniform baseline. The systematic grid evaluation of non-adaptive schedules serves as a second baseline. It allows to judge the performance of the algorithm relative to the possible gain that could be expected from *any* population size adaptation algorithm.

We consider the benchmark problems ZDT1-4 and ZDT6 from [16], the two-objective versions of DTLZ1-4 from [5], as well as Schaffer's problem. We use 30 variables for ZDT1-3 and 10 variables for all other problems. The goal of

optimization is to cover 99.9% of the hypervolume theoretically achievable with 100 individuals, relative to the reference point (1.1, 1.1). Visual inspection reveals that this formalized goal corresponds to a reasonably accurate problem solution. For the ZDT and DTLZ problems, the achievable hypervolume can be obtained from the website <http://www.tik.ee.ethz.ch/~sop/download/supplementary/testproblems/> for the reference point (11, 11) and converted easily. For Schaffer's problem we use the formulation $f_1(x) = |x_1 - \frac{1}{2}|$ and $f_2(x) = |x_1 + \frac{1}{2}|$, so that the optimal front fits inside the unit square. The achievable hypervolume with N points is $1.1^2 - 0.5 - 0.5/(N - 1)$.

The first experiment compares SMS-EMOA with population size 100 to the same algorithm with increasing population size schedules as described in section 3. We have tested a four-dimensional grid of schedules given by the parameters $\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5\}$, $\beta \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, and $\delta \in \{0.0, 0.1, 0.2, 0.3, 0.5, 0.7\}$. The budget FE_{budget} was set to the number of FE required by the baseline, rounded up, see table 1. The algorithm was run 100 times for each of the 1 050 schedules. The median number of FE relative to the baseline is reported compactly in figure 2.

The results show two basic facts. First, the region where a good schedule is found varies from problem to problem. This clearly shows the need for an adaptive strategy. Second, problems exist where the baseline is hard to beat with any schedule. This means that increasing populations help in many cases, but not always, while it (nearly) never harms.

On the DTLZ4 benchmark all strategies with initial population size of less than 50 hit the maximum of 500 000 FE *in the median*. It turns out that this result is not due to an algorithmic flaw but must be attributed solely to numerical problems².

In the second experiment, the population size online adaptation procedure is tested against plain SMS-EMOA and also ranked relative to the extensive grid of deterministic schedules. The experimental setup remains unchanged. The adaptive schedule has a single parameter α controlling the initial population size S_{start} . For a fair comparison, this parameter was varied in the same range as before.

Table 1. Budgets for the definition of the fixed schedules. The budget values were determined by rounding up the median FE required by plain SMS-EMOA for reaching 99.9% of the optimal hypervolume.

Problem	Schaffer	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DTLZ1	DTLZ2	DTLZ3	DTLZ4
Budget	6 000	9 000	11 000	10 000	40 000	7 000	36 000	6 000	105 000	6 000

² The term $\cos(\pi/2 \cdot x_i^{100})$ in the DTLZ4 problem applied to numbers $x_i < 0.83$ gives exactly one when evaluated with 64bit IEEE double precision numbers. This leads to a large fitness plateau and a spurious local optimum. Control experiments with (a) higher precision, (b) lower exponent, and (c) larger population confirm this finding. Therefore, we do not consider the DTLZ4 benchmark any longer.

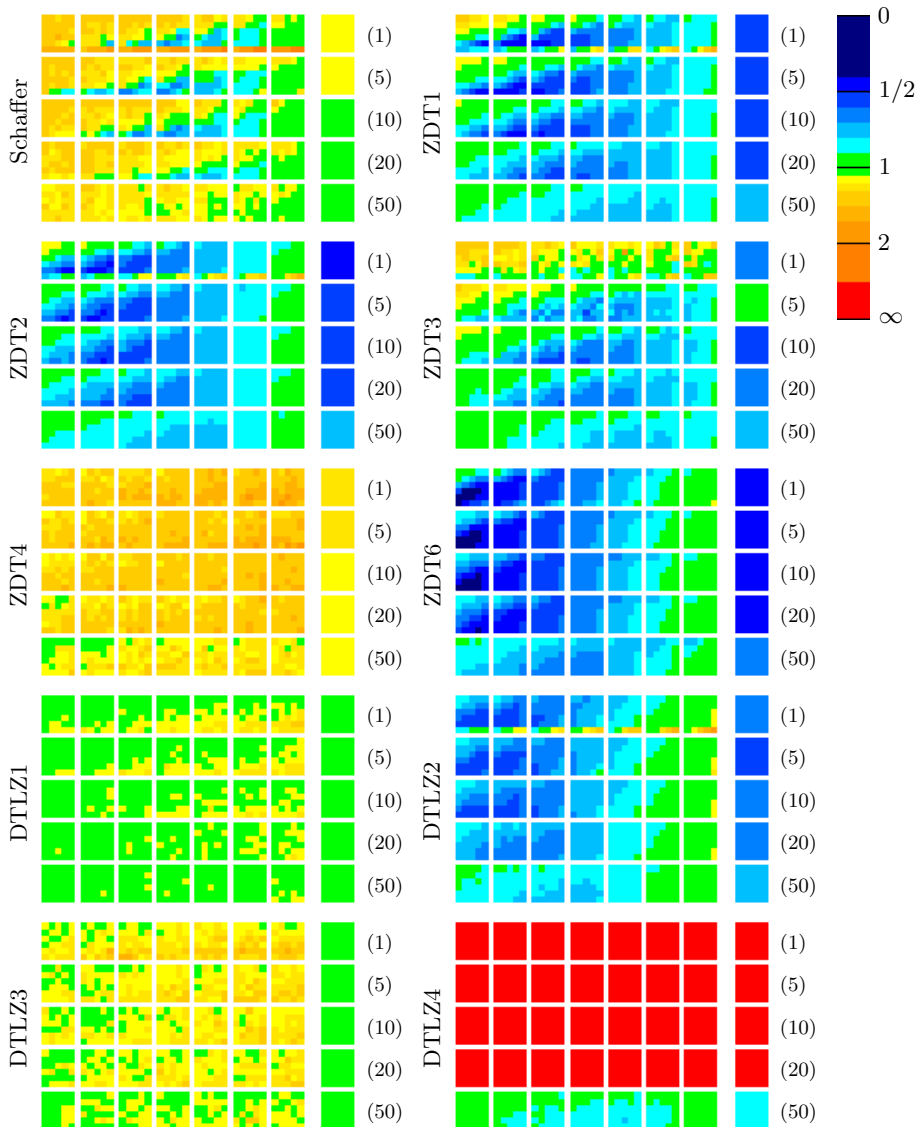


Fig. 2. Performance of fixed and adaptive schedules. Each 5×7 matrix of bitmaps encodes values of the parameters α (rows) and β (columns). In addition the initial population size is listed in brackets on the right. Within each bitmap, rows and columns encode parameters δ and γ , respectively, so that the position within the bitmap resembles the position of the intermediate point as indicated in figure 1. The column on the right of the matrix reports results for the adaptive schedule. Pixel colors indicate relative runtime, measures as number of FE divided by number of FE required by SMS-EMOA. Values smaller than 1 (blueish color) indicate an improvement over the baseline, values close to 1 (green) mark performance indifferent to the baseline, and values larger than 1 (yellow to red) indicate deterioration of the performance as compared to the baseline.

Table 2. Performance (median number of *FE*, lower is better) of plain SMS-EMO and adaptive population size schedule, with the same initial population size of $S_{\text{start}} = 10$

Problem	Schaffer	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DTLZ1	DTLZ2	DTLZ3
Baseline	5 482	8 867	10 271	9 533	39 422	6 012	35 052	5 015	102 185
Adaptive	5 848	4 851	5 544	5 712	46 591	2 701	34 926	3 058	97 880

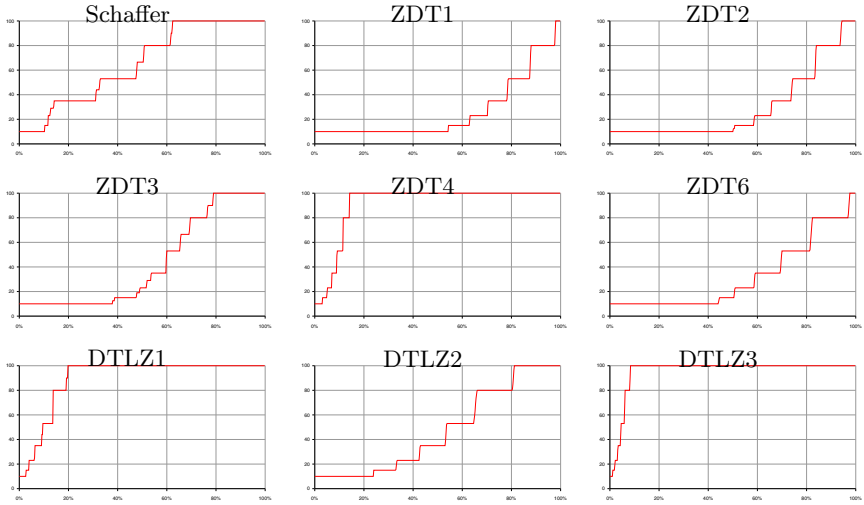


Fig. 3. Evolution of the adaptively controlled population size starting with 10 individuals. All successful runs were rescaled to the same length which is displayed on a percentage scale.

The results for the adaptive schedules are reported graphically in the column right to the bitmap matrix in figure 2, as well as numerically in table 2. The performance of the algorithm is rather robust w.r.t. its only parameter, the initial population size. Our results suggest a default setting of $\alpha = 0.1$ (corresponding to $S_{\text{start}} = 10$ in our experiments).

It becomes clear that on average the adaptive schedule works about as well as a good fixed schedule. Importantly, this is achieved across different problems that require different types of schedules, without prior knowledge of the budget, and with a practically parameter free method.

An interesting question is how exactly the online adaptation evolves the population size. Figure 3 answers this question. There is a high correlation between the population size staying low for an extended period and a significant performance improvement over the baseline (compare to figure 2). This is not surprising since increasing the population size quickly basically means that plain SMS-EMOA takes over quickly. Such behavior is hard to avoid on multi-modal problems such as ZDT4, DTLZ1, and DTLZ3. Importantly, although in these cases online adaptation does not help, it also does not (seriously) impair performance.

In contrast, for problems ZDT1, ZDT2, ZDT3, ZDT6, and DTLZ2 adaptively increasing the population size results in considerable savings of *FE*, in the order of about 50%. All of these problems can be solved by approaching the front with a small population, resulting in increased selection pressure, and spreading the increasing population over the front as soon as the progress rate drops.

In summary, starting small and growing the population big over the course of a multi-objective optimization run can save a significant fraction of fitness evaluations, while it nearly never hurts. Our adaptive algorithm performs in most cases about as well as the (in general unknown) best deterministic schedule.

5 Conclusion and Outlook

We have proposed an online adaptation scheme for the population size of an EMOA. This algorithm was compared with the usual proceeding of fixing the population size to the desired cardinality of the result set, as well to a large number of systematically chosen deterministic increasing population size schedules. The proposed adaptive algorithm compares favorably. It saves up to about 50% of the fitness evaluations of the standard algorithm in case uni-modal problems, whereas it shows nearly unchanged behavior on multi-modal benchmarks. The comparison of the performance of the adaptive schedule to the large set of deterministic schedules reveals that significantly better results cannot be expected with *any* population size schedule. Thus the strategy to *start small and grow big* turns out to be successful in MCO.

A few open questions remain for future research. We did not present an adaptation rule for shrinking of the population size, although a similar success-based adaptive rule is straightforward to design. However, at least on standard benchmarks shrinking is not very useful. Another open question is how the adaptation rule can be adapted to work with an evolution strategy without interfering with the (often success-based) step size adaptation mechanism. An extension of this study to more than two objectives is work in progress.

References

1. Bader, J., Zitzler, E.: HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation* 19(1), 45–76 (2011)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181(3), 1653–1669 (2007)
3. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd edn. Springer, New York (2007)
4. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, Chichester (2001)
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: *Congress on Evolutionary Computation, CEC 2002*, pp. 825–830. IEEE Press, Piscataway (2002)

6. DeJong, K.A.: *Evolutionary Computation: A Unified Approach*. MIT Press, Cambridge (2006)
7. Eiben, A.E., Smith, J.E.: *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, Berlin (2003)
8. Eskandari, H., Geiger, C.D., Lamont, G.B.: FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) *EMO 2007*. LNCS, vol. 4403, pp. 141–155. Springer, Heidelberg (2007)
9. Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation* 15(1), 1–28 (2007)
10. Laumanns, M.: *Analysis and Applications of Evolutionary Multiobjective Optimization Algorithms*. Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2003)
11. Lu, H., Yen, G.G.: Dynamic population size in multiobjective evolutionary algorithms. In: *Proceedings of the 2002 IEEE Congress on Evolutionary Computation, CEC 2002*, pp. 1648–1653. IEEE Press, Piscataway (2002)
12. Peschel, M., Riedel, C.: Use of vector optimization in multiobjective decision making. In: Bell, D.E., Keeney, R.L., Raiffa, H. (eds.) *Conflicting Objectives in Decisions*, pp. 97–121. Wiley, Chichester (1977)
13. Rudolph, G., Agapie, A.: Convergence properties of some multi-objective evolutionary algorithms. In: Zalzala, A., et al. (eds.) *Proceedings of the 2000 Congress on Evolutionary Computation, CEC 2000*, vol. 2, pp. 1010–1016. IEEE Press, Piscataway (2000)
14. Tan, K.C., Lee, T.H., Khor, E.F.: Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 5(6), 565–588 (2001)
15. Zitzler, E.: *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (November 1999)
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation* 8(2), 173–195 (2000)
17. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) *PPSN V*. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)

Queued Pareto Local Search for Multi-Objective Optimization

Maarten Inja, Chiel Kooijman, Maarten de Waard,
Diederik M. Roijers, and Shimon Whiteson

Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands
{maarten.inja, chiel.kooijman, mrtndwrd}@gmail.com,
{d.m.roijers, s.a.whiteson}@uva.nl

Abstract. Many real-world optimization problems involve balancing multiple objectives. When there is no solution that is best with respect to all objectives, it is often desirable to compute the *Pareto front*. This paper proposes *queued Pareto local search* (QPLS), which improves on existing *Pareto local search* (PLS) methods by maintaining a queue of improvements preventing premature exclusion of dominated solutions. We prove that QPLS terminates and show that it can be embedded in a genetic search scheme that improves the approximate Pareto front with every iteration. We also show that QPLS produces good approximations faster, and leads to better approximations than popular alternative MOEAs.

1 Introduction

Many real-world optimization problems contain multiple objectives, e.g., when mapping different processes of a software application to hardware components, both processing time and power consumption should be minimized [7].

For specialized applications, it is sometimes possible to compute an exact set of optimal trade-offs between the objectives, i.e., the Pareto front. However, when the solution space of these problems is large, computing the Pareto front is often intractable. In this case, *multi-objective evolutionary algorithms* (MOEAs) [2] can compute a set of solutions that approximates the Pareto front. MOEAs are general-purpose multi-objective optimization methods, and have been applied to a large variety of optimization problems [1], from reinforcement learning [15] to design space exploration [12].

To speed up MOEAs, one can use *Pareto local search* (PLS) algorithms [5,9,13]. PLS algorithms employ simple heuristic improvement algorithms to find reasonably good solutions in a short time. PLS methods find an approximate Pareto front by looking in the (search space) neighborhood of the individual solutions in this archive for solutions that improve upon the current Pareto archive. An important advantage of this method is that the size of the archive is not limited. This is important because the size of the Pareto front is often not known in advance.

However, an important downside of current PLS methods is that promising solutions are excluded from the Pareto-archive when an improvement is found for another (unrelated) solution, before they have a chance to improve themselves.

Such premature deletions from the archive are undesirable since they reduce genetic diversity that might be needed to find part of the Pareto front.

In this paper, we propose a new algorithm for Pareto local search that we call *queued Pareto local search* (QPLS), which stores promising solutions in a queue. When such a solution is popped off the queue, QPLS performs strict Pareto improvements by looking for a strictly better solution (in all objectives) in the neighborhood of the solution. Only when such improvements are no longer possible, is it compared to a Pareto archive. This prevents premature deletion of solutions. Unlike other PLS methods, the queued approach “protects” dominated solutions until they are mutated to a locally optimal state. As in previous PLS work, we embed QPLS in a genetic scheme.

We empirically compare Genetic QPLS against the state-of-the-art PLS method Genetic SPLS [5], and against the popular non-PLS MOEAs NSGA-II [3] and SPEA2 [21], using multi-objective coordination graphs [16,17]. We show that QPLS can produce good approximate fronts for much larger problems than that can be solved with exact methods (such as [18]). Furthermore, we show that Genetic QPLS maintains a Pareto set with a larger *hypervolume* [19] than the other algorithms both in the short and long run.

2 Background

We first introduce the problem setting and notation used throughout the paper. A *multi-objective optimization problem* (MOOP) is a tuple $\langle \mathcal{V}, \mathcal{S}, \mathbf{f} \rangle$, where:

- $\mathcal{V} = \{v_1, \dots, v_n\}$ is the set of n enumerated *variables*,
- $\mathcal{S} = \mathcal{S}_1 \times \dots \times \mathcal{S}_n$ is the *search space*, i.e., the Cartesian product of the domains of the variables in \mathcal{V} , and
- $\mathbf{f} : \mathcal{S} \rightarrow \mathbb{R}^d$ is a d -objective fitness function that maps all points in the search space to a d -objective fitness. We refer to the value of the i -th objective as f_i .

We define the neighborhood of s , $\mathcal{N}(s)$, as the set of solutions that differ in exactly one variable. We assume that each objective needs to be maximized. We refer to an approximation to the Pareto front as a *Pareto archive*, denoted P . A solution s *Pareto dominates* a solution s' when its fitness is larger in one objective, and at least equal in all the others: $\mathbf{f}(s) \succ \mathbf{f}(s') = ((\forall i) f_i(s) \geq f_i(s')) \wedge ((\exists j) f_j(s) > f_j(s'))$. In a Pareto archive, there are no solutions that are Pareto dominated by another solution in that set. We say that a solution s *weakly dominates* another solution s' , when it either dominates it, or has equal value. If s does not dominate s' and s' does not dominate s then s and s' are *incomparable*. Two Pareto archives P and P' can be *merged* [5] by taking the union and removing the dominated solutions.

2.1 Non-PLS Methods

Many MOEAs (e.g., [3,21]) do not employ PLS but instead maintain a population of individual solutions that are improved upon via mutation and crossover

operators. When these operators are ergodic, the entire search space is scanned in the limit [6]. This guarantees that MOEAs do not get permanently stuck in local optima. This holds even for MOEAs that do not employ PLS.

Two popular MOEAs that do not employ PLS are NSGA-II [3] and SPEA2 [21]. NSGA-II applies elitism, uses a fast sorting algorithm and focuses on maintaining diversity within the population. SPEA2 employs a measure of *strength* for a solution based on the number of solutions that dominate it, the number of solutions that are dominated by it, and the distance to the k -th nearest neighbor. The advantage of both is that their population diversity prevents them from getting stuck in local optima. However, neither of these algorithms employ heuristics to find better solutions more quickly. NSGA-II and SPEA2 are the two most popular algorithms for MOOPs [20].

2.2 PLS Methods

Paquete et al. [13] propose a *Pareto local search* (PLS) algorithm that maintains an archive of non-dominated solutions P . Each solution $s \in P$ is visited to search its neighborhood $\mathcal{N}(s)$ for new solutions. This is achieved by merging the entire neighborhood of s into P . A drawback is that the neighborhood of one solution can overwrite the whole archive (especially early on), while the neighborhoods of the overwritten solutions can contain solutions that would improve P even further. Drugan and Thierens [5] propose a variation in which improvements are made to the archive by adding a single improving solution with respect to P (Pareto-dominant or Pareto-incomparable) from the neighborhood $\mathcal{N}(s)$ of one solution $s \in P$ at a time. This reduces the probability that promising solutions in P are deleted after an improvement from a different solution. However, the removed dominated solutions might still have had possible improvements.

Liefooghe et al. [9] generalize PLS algorithms into several categories based on the current set selection for neighborhood scanning, exploration strategy, archiving method and stopping conditions. Two types of archiving methods are considered. An *unbounded* archive can be used to store the set of all non-dominated solutions and a *bounded* archive stores a subset of the non-dominated solutions. Some algorithms only store the dominating solutions as the archive is filled. Other algorithms employ diversity criteria, such as crowding distance or ε -dominance, to limit the size of the archive. However, all of these algorithms generate improvements from the neighborhood of current elements of the archive and directly delete members of the archive due to these improvements, even though these deleted elements could yield improvements if their neighborhoods were inspected.

Two distinct strategies can be employed to select the next improving solution s' from the neighborhood of s , $\mathcal{N}(s)$. In the *best-improvement* strategy the entire neighborhood $\mathcal{N}(s)$ is scanned for the best improvement. The selected improvement s' is guaranteed not to be dominated by another other solution in $\mathcal{N}(s)$. In the *first-improvement* strategy, the first solution s' in $\mathcal{N}(s)$ that improves s is returned immediately. Hansen and Mladenović [8] find that for single-objective LS, first-improvement usually leads to better empirical results. Liefooghe et al. [9] confirm this result for PLS algorithms, as do Drugan and Thierens [5] for SPLS.

3 QPLS

Our main contribution, *queued Pareto local search*, is given in Algorithm 1. QPLS prevents the premature deletion of promising solutions by maintaining a queue of solutions to improve, which leads to a more diverse Pareto archive.

Algorithm 1. QPLS(\mathbf{f} , Q , k)

Require: Initial queue Q

- 1: \blacktriangledown the Pareto front
- 2: $P \leftarrow \emptyset$
- 3: **while** $Q.\text{notEmpty}()$ **do**
- 4: $s \leftarrow Q.\text{pop}()$
- 5: \blacktriangledown recursive local improvements
- 6: $s \leftarrow \text{PI}(s, \mathbf{f})$
- 7: \blacktriangledown s undominated by P
- 8: **if** $\forall p \in P : \mathbf{f}(s) \not\prec \mathbf{f}(p) \wedge \mathbf{f}(s) \neq \mathbf{f}(p)$
- 9: $P \leftarrow \text{merge}(P, \{s\})$
- 10: \blacktriangledown new candidates
- 11: $N \leftarrow \{s' \in \mathcal{N}(s) : s \not\prec s'\}$
- 12: $Q.\text{addK}(N, k)$
- 13: **end if**
- 14: **end while**
- 15: **return** P

Algorithm 2. GQPLS(α , p_M)

Require: A random initial Q

- 1: $P \leftarrow \text{QPLS}(Q, \mathcal{I})$
- 2: **while** NOT Stopping condition **do**
- 3: $Q.\text{clear}()$
- 4: **for** $s \in P$ **do**
- 5: **if** $\alpha > U(0, 1)$ or $|P| < 2$
- 6: $s' \leftarrow \text{Mutate}(s, p_M)$
- 7: **else**
- 8: Select $s_1 \neq s$ from P
- 9: $s' \leftarrow \text{Recombine}(s, s_1)$
- 10: **end if**
- 11: $Q.\text{add}(s')$
- 12: **end for**
- 13: $P \leftarrow \text{merge}(P, \text{QPLS}(Q))$
- 14: **end while**
- 15: **return** P

The algorithm starts with an initial queue Q of candidate solutions, and an empty Pareto archive (line 2). Until the queue is empty, these candidate solutions are popped one by one (line 3–14). When a solution s is obtained from the queue, it first runs a recursive *Pareto improvement* function (PI) at line 6. PI improves solutions by repeatedly selecting a dominating solution from the neighborhood, until such improvements are no longer possible.

After a solution s is found that is not dominated by any of its neighbors, it is compared to the Pareto archive P (line 8), and when it is not weakly dominated by any solution in P , s is merged into P (line 9). Then, new candidate solutions are selected from the neighbors of s . The set N on line 11 is the set of neighbors of s that are incomparable to s . From N , k candidates are randomly selected to be added to the queue. By setting the parameter k , the amount of exploration in the neighborhood of one solution can be controlled. Making k too large leads to exploring a lot of unsuccessful candidates initially, whereas making k too small reduces the archive’s genetic diversity. We typically choose k between 2 and 10.

Following previous work [5,8], we can distinguish two strategies for PI, a *best-improvement* and a *first-improvement* implementation. Unlike SPLS, where improvements are applied once per iteration, we apply the improvements recursively until no improvement can be found. Because solutions that have not yet been optimized are protected in the queue, we do not have to worry about premature deletions from our intermediate Pareto archive.

QPLS guarantees that all solutions in the Pareto archive are Pareto-undominated with respect to their neighborhoods at any given time while the

algorithm runs. Furthermore, it can be proved that, in a finite state space, QPLS terminates in a finite number of steps for any finite initial queue. This is because there can be only a limited number of steps to any archive P^* , and because there can be no cycles in its execution.

However, QPLS converges to a locally optimal set. A naive method to find better approximate Pareto fronts than the single locally optimal set returned by QPLS is just to restart QPLS with different random initial queues, and *merge* the result, i.e., *multi-start QPLS* (MQPLS). However, MQPLS does not exploit the results of the individual QPLS runs to focus on more promising regions.

4 Genetic QPLS

Genetic QPLS (GQPLS) escapes local optima by mutating and recombining the entire Pareto archive that result from individual QPLS runs. After a single QPLS run, we can mutate all the solutions in the archive, and restart PLS with these mutated solutions as input. In the case of QPLS, this input is the initial queue.

The *genetic local search* (GLS) scheme combines mutation and recombination. We define GQPLS in Algorithm 2. It escapes local optima by restarting QPLS with a set of new solutions, consisting of mutations and recombinations of solutions from Pareto archive P returned by single QPLS runs. The implementation of the genetic operators *Mutate* and *Recombine* are dependent on the problem.

GQPLS starts by running QPLS on an initial queue Q to find a locally optimal Pareto archive P on line 1. On lines 3 to 12, a new queue is created which consists of mutations and recombinations from the previous archive. For each solution $s \in P$, either (with probability α) a mutation of s is generated (line 6), or (with probability $1 - \alpha$) we use a recombination of s with a random of solution $s' \neq s$ from the archive (line 9). The newly generated solution is then added to the new queue. Finally, QPLS is called again with the new queue, and the result of which is merged into P . Because we use the Pareto dominance relationship in *merge*, the archive can only improve or remain the same. GQPLS runs indefinitely or until some stopping condition is met.

When the crossover probability $\alpha = 0$, one could regard this as an *iterated* [4,10] version of QPLS. We view iterated QPLS as a special case of GQPLS.

5 Experiments

We compare QPLS to existing PLS and non-PLS MOEAs on randomly generated *multi-objective coordination graphs* (MO-CoGs) [16], which are single-state problems from the multi-agent literature in which agents must work together in order to obtain a shared (vector-valued) reward. Not only do these problems form an important problem on their own (e.g., for resource gathering [16] or risk-sensitive combinatorial auctions [11]), they are also a key subproblem in more elaborate sequential settings (such as transport network maintenance planning [14]). In these settings, time is often limited, making fast heuristic methods key

to their applicability. Additionally, MO-CoGs allow for fast evaluation of a mutation of an evaluated solution, which makes it possible to perform faster local search. MO-CoGs form a class of flexible and scalable problems that can vary in size (the number of variables and the size of the search space) and in complexity (the complexity of the graphs), making them interesting MOEA benchmarks.

MO-CoGs are MOOPs in which \mathcal{V} is a set of n enumerated agents, $\mathcal{S} = \mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$ is the Cartesian product of the action spaces of the individual agents. A specific joint action is denoted \mathbf{a} . Finally, \mathbf{f} is a vector sum over a set of local payoff functions \mathcal{U} : $\mathbf{f}(\mathbf{a}) = \sum_{\mathbf{u}_e \in \mathcal{U}} \mathbf{u}_e(\mathbf{a}_e)$. A local payoff function \mathbf{u}_e has limited scope e , i.e., only a subset of agents participate in it. A local joint action of the agents that participate in \mathbf{u}_e is denoted \mathbf{a}_e . The local joint actions \mathbf{a}_e on the righthand side are derived from the full joint action \mathbf{a} on the lefthand side.

The local payoff functions and the agents can be seen as the two sets of nodes in a bipartite graph whose edges indicate which agents participate in which local payoff functions. A small example graph with 3 agents (circles) and 2 local payoff functions is shown in Fig. 1.

To generate MO-CoGs, we follow the random graph generation procedure proposed by Roijers et al. [16]. This procedure takes the following input variables: n , the number of agents; d , the number of objectives; ρ the number of local payoff functions; and $|\mathcal{A}_i|$, the action space size, which in this procedure is the same for all agents. The values in each local payoff function are filled with real numbers drawn independently from a uniform distribution on the interval $0 < u_{e,i}(\mathbf{a}_e) < 10$.

The payoff for a single local function depends on the actions of all connected agents, and each agent can participate in several local payoff functions. Therefore, getting good payoffs requires carefully coordinated actions. Randomly changing the action of random agents in a carefully balanced solution can therefore cause the coordination to collapse. To minimize the impact of changes in a solution but still be able to escape from local optima, the function **Recombine** copies the actions for adjacent agents from each solution. Specifically, it uses the graph structure to decide where it is divided: starting with a random payoff function, it adds the action from the first solution and iterates breadth-first over the agents in the graph, stopping with a probability p_S or when half of the graph has been covered. The other actions are taken from the second solution. p_S was chosen so that the expected number of actions changed is a third of the total.

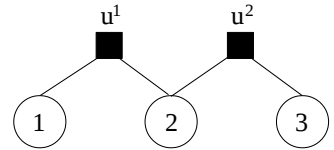


Fig. 1. A coordination graph

The **Mutate** operator, by contrast, does not take graph structure into account. It returns a new solutions that is a copy of its argument, except that every action has a probability $p_M = 0.05$ to switch to a random action in its action space.

We compare against two non-PLS-employing MOEAs: NSGA-II and SPEA2, and against the PLS-employing method we empirically found best for MO-CoGs: Drugan and Thierens' first-improvement genetic SPLS [5]. We employ the same **Mutate** and **Recombine** operators described above for all methods. All algorithms,

as well as the problem, were implemented in JAVA. All experiments were run on an Intel Core i7-3610QM quad core CPU at 2.30GHz.

5.1 Parameter Optimization

For MO-CoGs of varying size, we optimized the parameters for each algorithm separately. For SPEA2 a low probability of crossover (0.2) was better than only using mutation. For NSGA-II 0.0 was best.

Genetic SPLS yielded the best results when the mutation probability α was set to 0.5. The first Pareto-improvement exploration strategy (as was also found by Drugan and Thierens [5] for quadratic assignment problems) was best.

Surprisingly, for Genetic QPLS the mutation only probability was best set to $\alpha = 1$. This can be explained by the observation that GQPLS has little trouble finding good solutions in the center of the Pareto front but more difficulty finding them around the edges of the Pareto front (as shown Fig. 4, which we discuss further in Sect. 5.3). Recombining values from the center of the front may be less likely to yield results on the edges, where the combination of two solutions from different edges would result in a solution around the center.

For GQPLS, recursive best Pareto-improvement performed best. This contradicts findings for earlier PLS algorithms [9], including GSPLS, the best other PLS algorithm for this problem. This can be explained by the protection of candidate solutions in the queue of QPLS. Best-improvement takes bigger steps on average; therefore, in other PLS algorithms, the probability that promising solutions in the Pareto archive are dominated by the resulting improvements and thus prematurely deleted, is higher as well. Because QPLS does not run the risk of deleting other candidate solutions from the queue, the best possible improvement strategy just speeds up the search.

The maximal numbers of additions to the queue in QPLS was best set to $k = 5$. When k was set higher, the individual runs of QPLS took longer, with only slightly better results (that were more easily achieved with the genetic scheme), and when k was lower the Pareto archives were initially narrower.

5.2 Approximation of the Pareto Front

We generated small MO-CoGs for which P^* was computed exactly using *multi-objective bucket elimination* (MO-BE) [18]. We generated 2-objective MO-CoGs of $n = 5$ to $n = 30$ agents, with $\rho = 1.5n$ and $|\mathcal{A}_i| = 10$. For larger MO-CoGs, it is not feasible to compute the true Pareto front, as MO-BE has a runtime that is exponential in the number of agents [16], and for more than 30 agents, its memory requirements become intractable.

Pareto-set approximations are evaluated in terms of the hypervolume [22] as a function of runtime. The hypervolume is defined as the volume of reward space that is dominated between the positive reward origin and the Pareto archive.

In a large portion of MOEA literature, methods are compared using the number of fitness evaluations instead of time. However, PLS methods do not do full fitness evaluations quite as often. Instead, they perform many tiny changes on a local level, which can be evaluated in a fraction of the time, and only evaluate fitness fully after recombinations and mutations. As a result, comparing the

number of fitness evaluations gives an incomplete and possibly misleading picture of the computational costs. Hence, we measure hypervolume as a function of runtime rather than number of evaluation function calls.

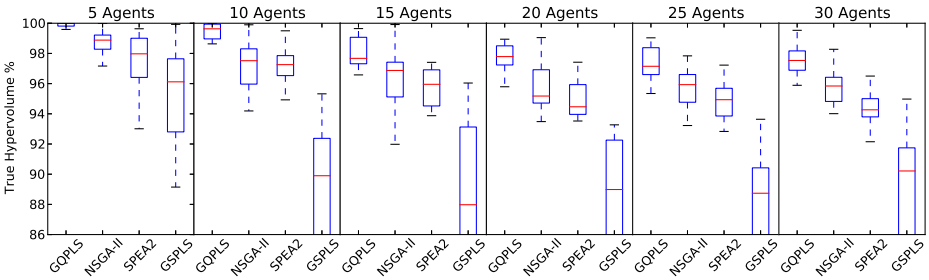


Fig. 2. Percentage of the real hypervolume found by GQPLS, NSGA-II, SPEA2 and GSPLS after 15 minute runs on MO-CoGs of varying size; 10 runs per problem size

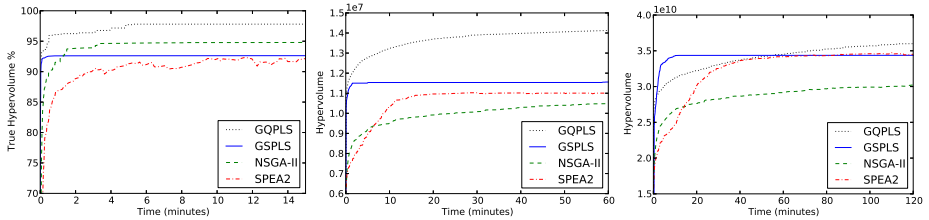


Fig. 3. The hypervolume as a function of time for one randomly generated MO-CoG for all tested MOEAs. (left) An $n = 30$, $d = 2$ MO-CoG, with hypervolume as a percentage of the hypervolume of the true Pareto front. (middle) An $n = 300$, $d = 2$ MO-CoG. (right) An $n = 300$, $d = 3$ MO-CoG.

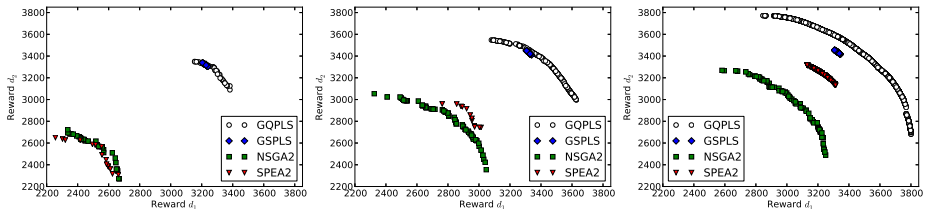


Fig. 4. The Pareto-archives found by GQPLS, GSPLS, NSGA-II and SPEA2 on a $n = 300$ MO-CoG: (left) after 20 seconds, (middle) 5 minutes, and (right) 1 hour

Figure 2 shows the fraction of the hypervolume of P^* that is found by the MOEAs within 15 minutes in quantiles. All methods produce less accurate approximations as the problem size increases. However, GQPLS performs better in terms of best, worst, and mean results than the other methods.

Figure 3 (left) shows the growth of the hypervolume over time for an $n = 30$ MO-CoG. After 15 minutes, all methods are still improving their Pareto fronts but GQPLS has better approximations than the other algorithms at every timestep. When comparing GQPLS to MO-BE in the 30-agent setting, we found that after one iteration, which on average took 0.37% of the time required by

MO-BE to calculate the true Pareto front, GQPLS found a Pareto archive that covered 92.4% of the hypervolume of P^* .

We conclude that GQPLS can approximate P^* better and more quickly than GSPLS, NSGA-II and SPEA2 for small problems, and that the difference in approximation quality tends to get bigger as the problem size increases. Furthermore, GQPLS provides good results in a fraction of the time MO-BE takes to find P^* .

5.3 A Large MO-CoG

One of the strengths of MOEAs lies in the fact that they can return approximate results when calculating the exact Pareto front is intractable. Figure 3 shows the results for $n = 300$, $d = 2$ (middle) and $n = 300$, $d = 3$ (right). In the $d = 2$ problem, GQPLS outperforms the other algorithms by a large margin, while the difference is smaller in the $d = 3$ problem. A likely explanation is that the benefit of protecting diversity (by using the queue) provides a smaller benefit in higher dimensional problems, as there are more ways in which solutions can be Pareto-equivalent, and thus fewer solutions are dominated and discarded.

Figure 4 shows how the fronts develop over time. Both NSGA-II and SPEA2 have a wide spread from early on and move slowly towards a higher reward in both dimensions. Both GSPLS and GQPLS improve towards the center of the graph first, but where GSPLS has difficulty widening the front, GQPLS is able to explore the front's edges while also finding improvements in the center. We hypothesize that this is because improvements on the edges are often the result of mutations that first take a rather big step back in both objectives and only turn out well after a full recursive Pareto-improvement run. It is precisely these mutations that need to be protected from premature deletion until such improvement has taken place.

Overall, these results show that, like other PLS-based methods, GQPLS finds a relatively good approximate front quickly and continues to improve in order to find the best approximations. We therefore conclude that using a PLS method that employs queues to protect promising candidates during Pareto local search can lead to great speed-ups in MOEAs.

6 Discussion and Conclusion

In this paper, we proposed a PLS algorithm based on the principle that newly mutated solutions should be allowed to find their full potential before comparing them to other solutions. QPLS therefore protects these solutions in a queue. We embedded QPLS in a genetic search scheme, yielding Genetic QPLS, to escape from local optima. QPLS terminates and GQPLS finds the true Pareto front in the limit. We showed empirically that QPLS outperforms other popular evolutionary multi-objective algorithms on random generated multi-objective coordination graphs, where mutations of known solutions can be evaluated efficiently.

In future work, we aim to employ ideas from other successful algorithms, such as NSGA-II and SPEA2, and focus the search on those regions of the value space that are less “crowded” by other solutions. The crowding distance, as

defined by NSGA-II, can be used for such purposes. There are several ways the crowding distance could be employed: (1) embed it inside QPLS to let solutions with a higher crowding distance produce more candidate solutions in the queue, (2) use it inside recursive best Pareto improvement to discriminate between Pareto incomparable solutions, or (3) use it to seed new initial queues in the outer loop of GQPLS. Furthermore we would like to compare GQPLS to more recent MOEAs such as those reviewed by Zavala et al. [20].

Acknowledgements. This research is supported by the NWO DTC-NCAP (#612.001.109) project.

References

1. Coello Coello, C.A., Lamont, G.B.: Applications of multi-objective evolutionary algorithms, vol. 1. World Scientific (2004)
2. Coello, C.A.C., Lamont, G.B., Van Veldhuisen, D.A.: Evolutionary algorithms for solving multi-objective problems. Springer, Heidelberg (2007)
3. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN VI. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
4. Drugan, M.M., Thierens, D.: Path-guided mutation for stochastic Pareto local search algorithms. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 485–495. Springer, Heidelberg (2010)
5. Drugan, M.M., Thierens, D.: Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies. *J. Heur.* 18(5), 727–766 (2012)
6. Eberhart, R.C., Shi, Y.: Comparison between genetic algorithms and particle swarm optimization. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 611–616. Springer, Heidelberg (1998)
7. Erbas, C., Cerav-Erbas, S., Pimentel, A.D.: Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. *IEEE Transactions on Evolutionary Computation* 10(3), 358–374 (2006)
8. Hansen, P., Mladenović, N.: First vs. best improvement: An empirical study. *Discrete Appl. Math.* 154(5), 802–817 (2006)
9. Liefvooghe, A., Humeau, J., Mesmoudi, S., Jourdan, L., Talbi, E.-G.: On dominance-based multiobjective local search: Design, implementation and experimental analysis on scheduling and traveling salesman problems. *J. Heur.* 18(2), 317–352 (2012)
10. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. arXiv preprint math/0102188 (2001)
11. Marinescu, R.: Efficient approximation algorithms for multi-objective constraint optimization. In: Brafman, R. (ed.) ADT 2011. LNCS (LNAI), vol. 6992, pp. 150–164. Springer, Heidelberg (2011)
12. Obayashi, S., Jeong, S., Chiba, K.: Multi-objective design exploration for aerodynamic configurations. AIAA, 4666:2005 (2005)
13. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Gandibleux, X., Sevaux, M., Sörensen, K., T’kindt, V. (eds.) Metaheuristics for Multiobjective Optimisation. LNEMS, vol. 535, pp. 177–199. Springer, Heidelberg (2004)

14. Roijers, D.M., Scharpff, J., Spaan, M.T., Oliehoek, F.A., de Weerd, M., Whiteson, S.: Bounded approximations for linear multi-objective planning under uncertainty. In: ICAPS (2014)
15. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. *JAIR* 47, 67–113 (2013)
16. Roijers, D.M., Whiteson, S., Oliehoek, F.A.: Computing convex coverage sets for multi-objective coordination graphs. In: Perny, P., Pirlot, M., Tsoukiàs, A. (eds.) ADT 2013. LNCS, vol. 8176, pp. 309–323. Springer, Heidelberg (2013)
17. Roijers, D.M., Whiteson, S., Oliehoek, F.A.: Linear support for multi-objective coordination graphs. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, pp. 1297–1304. International Foundation for Autonomous Agents and Multiagent Systems (2014)
18. Rollón, E., Larrosa, J.: Bucket elimination for multiobjective optimization problems. *Journal of Heur.* 12(4-5), 307–328 (2006)
19. Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., Dekker, E.: Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Mach. Learn.* 84(1-2), 51–80 (2011)
20. Zavala, G., Nebro, A., Luna, F., Coello Coello, C.A.: A survey of multi-objective metaheuristics applied to structural optimization. *Struct. Multidiscip. O.*, 1–22 (2013)
21. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm (2001)
22. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)

Distance-Based Analysis of Crossover Operators for Many-Objective Knapsack Problems

Hisao Ishibuchi, Yuki Tanigaki, Hiroyuki Masuda, and Yusuke Nojima

Department of Computer Science and Intelligent Systems, Graduate School of Engineering,
Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
{hisaoi@,yuki.tanigaki@ci.,hiroyuki.masuda@ci.,nojima@}
cs.osakafu-u.ac.jp

Abstract. It has been reported for multi-objective knapsack problems that the recombination of similar parents often improves the performance of evolutionary multi-objective optimization (EMO) algorithms. Recently performance improvement was also reported by exchanging only a small number of genes between two parents (i.e., crossover with a very small gene exchange probability) without choosing similar parents. In this paper, we examine these performance improvement schemes through computational experiments where NSGA-II is applied to 500-item knapsack problems with 2-10 objectives. We measure the parent-parent distance and the parent-offspring distance in computational experiments. Clear performance improvement is observed when the parent-offspring distance is small. To further examine this observation, we implement a distance-based crossover operator where the parent-offspring distance is specified as a user-defined parameter. Performance of NSGA-II is examined for various parameter values. Experimental results show that an appropriate parameter value (parent-offspring distance) is surprisingly small. It is also shown that a very small parameter value is beneficial for diversity maintenance.

Keywords: Mating schemes, evolutionary multiobjective optimization (EMO), many-objective optimization, knapsack problems, NSGA-II.

1 Introduction

Evolutionary multi-objective optimization (EMO) has been an active research area in the field of evolutionary computation in the last two decades. A number of multi-objective continuous optimization problems have been proposed as test problems in the EMO community. Whereas continuous problems have been mainly used to evaluate the performance of EMO algorithms, combinatorial test problems such as multi-objective knapsack problems in Zitzler and Thiele [17] have also been used (e.g., see Jaszkiwicz [9], Sato et al. [13], and Zhang and Li [16]).

For multi-objective knapsack problems, it has been reported in some studies [3], [6], [13] that the recombination of similar parents improves the performance of EMO algorithms such as SMS-EMOA [1] and NSGA-II [2]. MOEA/D [16] has an inherent mechanism of recombining similar parents, which is local selection of parents based on a neighborhood structure of solutions. It has been reported in [4] that the removal

of local selection deteriorates the performance of MOEA/D on multi-objective knapsack problems. Reported results in those studies suggest the existence of a negative effect of recombining totally different parents on the performance of EMO algorithms. Recently Sato et al. [14] demonstrated that the performance of EMO algorithms was improved by exchanging only a small number of genes between two parents (i.e., using a very small gene exchange probability) instead of choosing similar parents.

In this paper, we examine the above-mentioned two schemes for performance improvement of EMO algorithms through computational experiments on 500-item knapsack problems with 2-10 objectives. NSGA-II [2] is used to examine the effect of each scheme. That is, NSGA-II is applied to each test problem under three settings of crossover: Standard uniform crossover, uniform crossover of similar parents, and modified uniform crossover with a very small gene exchange probability. In computational experiments, we measure the parent-parent distance and the parent-offspring distance. Good results are obtained when the parent-offspring distance is small.

To further examine this observation, we implement a distance-based crossover operator where the generated offspring always has a pre-specified distance from its closer parent. That is, the parent-offspring distance is specified as a user-defined parameter. Performance of NSGA-II is measured for various parameter values to examine the relation between the parent-offspring distance and its performance. As performance measures, we calculate the hypervolume of solutions obtained from each run of NSGA-II using two reference points. One is far from and the other is close to the Pareto front. The two reference points are used to examine the effect of the parent-offspring distance on the diversification and convergence properties of NSGA-II.

The rest of this paper is organized as follows. In Section 2, we briefly explain our test problems (i.e., 500-item knapsack problems with 2-10 objectives). In Section 3, we explain the above-mentioned two performance improvement schemes. In Section 4, we report our experimental results where the performance of NSGA-II with each scheme is evaluated. In Section 5, we discuss our experimental results using a distance-based crossover operator. Finally we conclude this paper in Section 6.

2 Multi-Objective and Many-Objective Knapsack Problems

Multi-objective knapsack problems with 2-4 objectives and 250, 500 and 750 items were used in Zitzler and Thiele [17]. Their two-objective n -item problem is written as

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})), \quad (1)$$

$$\text{subject to } \sum_{j=1}^n w_{ij}x_j \leq c_i, \quad i = 1, 2, \quad (2)$$

$$x_j = 0 \text{ or } 1, \quad j = 1, 2, \dots, n, \quad (3)$$

$$\text{where } f_i(\mathbf{x}) = \sum_{j=1}^n p_{ij}x_j, \quad i = 1, 2. \quad (4)$$

In Zitzler and Thiele [17], the profit p_{ij} and the weight w_{ij} of item j with respect to knapsack i were specified as random integers in the interval [10, 100]. The capacity c_i of knapsack i was specified as 50% of the total weight of n items for knapsack i . Since the number of objectives and the number of items can be arbitrarily specified, multi-objective knapsack problems have been used as many-objective test problems (e.g., [4], [7], [14]). They have also been used as large-scale multi-objective test problems with up to 10000 items [5].

In this paper, we use multi-objective 500-item knapsack problems with 2-10 objectives. Those test problems are generated from the two-objective n -item problem in (1)-(4) by specifying n as $n = 500$ and creating additional objectives as follows:

$$f_i(\mathbf{x}) = \sum_{j=1}^n p_{ij}x_j, \quad i = 3, 4, \dots, 10, \quad (5)$$

where the profit p_{ij} of item j with respect to knapsack i is specified as a random integer in the interval [10, 100] in the same manner as in [17]. We denote the k -objective 500-item knapsack problem as the k -500 problem. In this paper, we use five test problems with $k = 2, 4, 6, 8, 10$ (i.e., 2-500, 4-500, 6-500, 8-500, 10-500 problems).

The constraint conditions in (2) and (3) are always used in our test problems independent of the number of objectives. This means that all of our test problems have the same set of feasible solutions. As a result, the same greedy repair method in [17] is used for constraint handling in all test problems in our computational experiments.

3 Two Performance Improvement Schemes

A similarity-based mating scheme was proposed and incorporated into NSGA-II to recombine similar parents in Ishibuchi et al. [6]. In its simplest version, first one parent is selected in the same manner as in NSGA-II (i.e., binary tournament selection with replacement based on non-dominated sorting and crowding distance). Next β candidates are selected by iterating the same parent selection mechanism as NSGA-II β times. Then the closest candidate to the first parent is selected from the β candidates using the Euclidean distance in the objective space. The selected candidate is used as the mate of the first parent. The standard uniform crossover operator is applied to the selected pair of similar parents. In the mating scheme, β is a user-defined parameter to specify the strength of the selection pressure toward similar parent selection. The larger value of β means the stronger selection pressure toward similar parent selection (i.e., stronger tendency to choose similar parents). When $\beta=1$, the mating scheme does not change the parent selection mechanism of NSGA-II at all.

Sato et al. [14] proposed an idea of exchanging only a small number of genes between two parents instead of selecting similar parents. They implemented the idea for uniform crossover by using a very small gene exchange probability, which was denoted by α_u in [14]. In the standard uniform crossover, genes of two parents are exchanged at each locus with the probability 0.5. In [14], good results were obtained for multi-objective knapsack problems when α_u was very small (e.g., 0.01).

4 Experimental Results

Three settings of crossover in NSGA-II are examined: Standard uniform crossover, uniform crossover of similar parents, and modified uniform crossover with a small gene exchange probability. One of the two children of crossover is randomly selected and handled as an offspring in NSGA-II in our computational experiments. The following parameter values are examined in the performance improvement schemes:

The number of candidates: $\beta = 1, 5, 10, 20, 30, 40, 50$.

Gene exchange probability: $\alpha_u = 0.01, 0.02, 0.03, 0.04, 0.05, 0.10, 0.15, 0.20$.

NSGA-II with each parameter value is applied to each test problem 100 times. The hypervolume of the obtained solutions is calculated using a fast calculation method [15] for each of the 100 runs for two reference points. One is $(0, 0, \dots, 0)$ which is far from the Pareto front, and the other is $(15000, 15000, \dots, 15000)$ which is close to the Pareto front. Computational experiments are performed under the following settings:

Population: 100 binary strings of length 500 with random initialization,

Termination condition: Evaluation of 400,000 solutions,

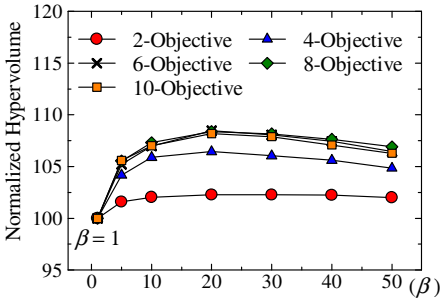
Crossover probability: 0.8 (One of the three versions of uniform crossover),

Mutation probability: 1/500 (Bit-flip mutation).

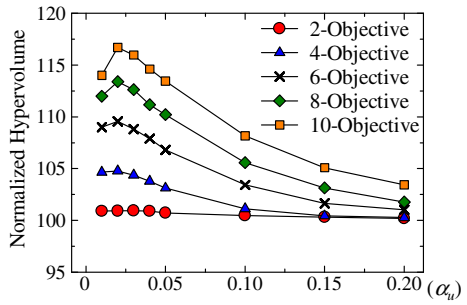
The calculated hypervolume value for each run with each parameter specification of β and α_u is normalized using the average result by NSGA-II with the standard uniform crossover for each reference point. It should be noted that the standard uniform crossover corresponds to the setting of $\beta = 1$ and $\alpha_u = 0.5$. In computational experiments, we also calculate the Hamming distance between two parents and between an offspring and its closer parent. The parent-offspring distance is measured after mutation only when crossover is used. Experimental results are summarized in Fig. 1 and Fig. 2 for each performance improvement scheme.

From Fig. 1 (a), we can see that the average normalized hypervolume value for the reference point $(0, 0, \dots, 0)$ is improved for all test problems by similar parent recombination from the baseline value 100 by the standard uniform crossover with $\beta = 1$ (the baseline value 100 is also obtained from the setting of $\alpha_u = 0.5$). In Fig. 2 (a), larger performance improvement is achieved for the 6-500, 8-500 and 10-500 problems by small gene exchange probabilities than similar parent recombination in Fig. 1 (a). However, better results are obtained for the 2-500 and 4-500 problems by similar parent recombination in Fig. 1 (a) than small gene exchange probabilities in Fig. 2 (a).

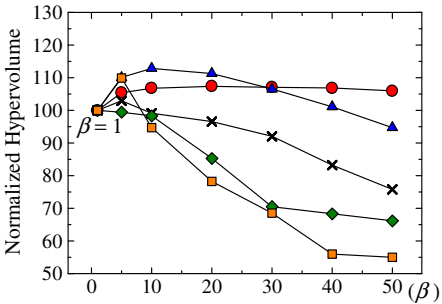
When the hypervolume for the reference point $(15000, 15000, \dots, 15000)$ is used as a performance measure in Fig. 1 (b) and Fig. 2 (b), we can observe both positive and negative effects of the performance improvement schemes. In Fig. 1 (b), the average normalized hypervolume value is improved by similar parent recombination for the 2-500 and 4-500 problems. However, the performance is degraded by similar parent recombination for the 6-500, 8-500 and 10-500 problems. In Fig. 2 (b), the use of too small gene exchange probabilities (e.g., $\alpha_u = 0.01$) severely degrades the performance for the 6-500, 8-500 and 10-500 problems. However, when α_u is specified between 0.1 and 0.2, the use of small gene exchange probabilities improves the performance for those test problems.



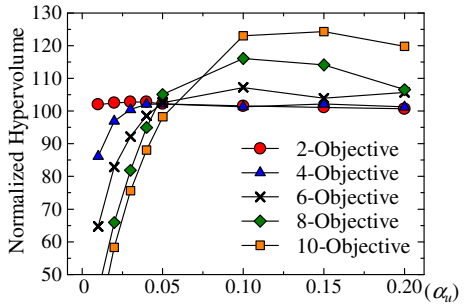
(a) Hypervolume for (0, 0, ..., 0)



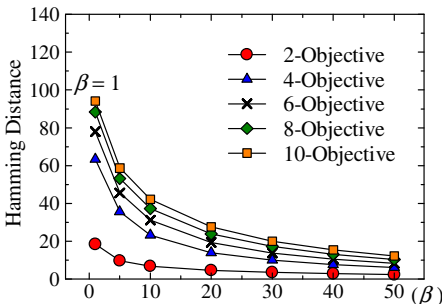
(a) Hypervolume for (0, 0, ..., 0)



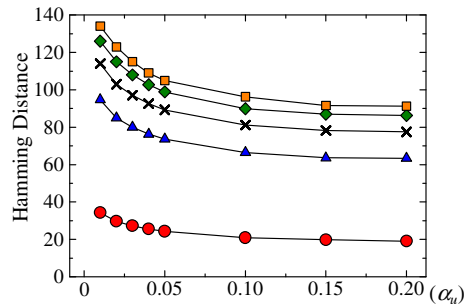
(b) Hypervolume for (15000, ..., 15000)



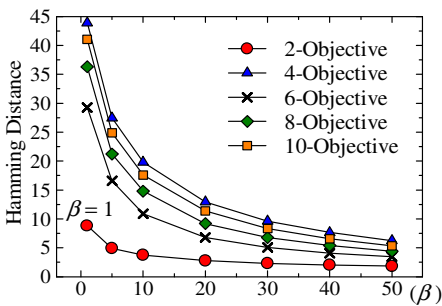
(b) Hypervolume for (15000, ..., 15000)



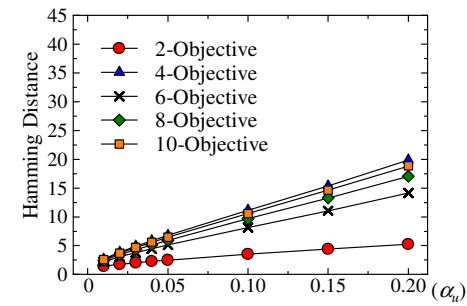
(c) Parent-parent Hamming distance



(c) Parent-parent Hamming distance



(d) Parent-offspring Hamming distance



(d) Parent-offspring Hamming distance

Fig. 1. Similar Parent Recombination

Fig. 2. Small gene exchange probability

Fig. 1 (c) and Fig. 2 (c) show the average Hamming distance of two parents while the distance in Fig. 1 (d) and Fig. 2 (d) is measured between an offspring and its closer parent. In Fig. 1 (c), the average Hamming distance between two parents is decreased by increasing the value of β . That is, the similarity of two parents is increased by increasing the value of β . As a result, the parent-offspring distance is decreased by increasing the value of β in Fig. 1 (d). We can also see from Fig. 1 (c) and Fig. 1 (d) that both the parent-parent distance and the parent-offspring distance are increased by increasing the number of objectives.

In Fig. 2 (c), the parent-parent distance is not small. This is because no selection mechanism of similar parents is used in Fig. 2. Thus, the average parent-parent distance in Fig. 2 (c) can be considered as being similar to the average distance between two solutions in a population over all generations (whereas they are not exactly the same). Fig. 2 (c) shows that the average parent-parent distance is increased by very small gene exchange probabilities for all test problems. This observation suggests that the performance improvement in Fig. 2 (a) is achieved by the increase in the diversity of solutions. At the same time, the diversity improvement severely degrades the convergence property, which leads to severe performance deterioration in Fig. 2 (b) by very small gene exchange probabilities for the 6-500, 8-500 and 10-500 problems.

In Fig. 1 (d) and Fig. 2 (d), the parent-offspring distance is decreased by increasing the parameter value of β (i.e., increasing the number of candidates for the second parents) and decreasing the parameter value of α_u (i.e., decreasing the gene exchange probability), respectively. From Fig. 1 (a) and Fig. 1 (d), we can see that good results are obtained in Fig. 1 (a) when the parent-offspring distance is small in Fig. 1 (d). The same observation is also obtained from Fig. 2 (a) and Fig. 2 (d).

5 Further Discussions Using Distance-Based Crossover

In this section, we further discuss the relation between the parent-offspring distance and the performance of NSGA-II. First we explain binary crossover using the concept of geometric crossover, which has been proposed by Moraglio and Poli [10]-[12]. Standard binary crossover (e.g., uniform, one-point, and two-point crossover) is geometric crossover in the sense that the sum of the Hamming distances between an offspring and its two parents is always equal to the Hamming distance between the two parents [8], [10]-[12]. That is, the following relation always holds for an offspring C generated by standard binary crossover from its two parents P_1 and P_2 :

$$H(C, P_1) + H(C, P_2) = H(P_1, P_2), \quad (6)$$

where $H(A, B)$ shows the Hamming distance between binary strings A and B . This relation always holds for all standard binary crossover operators [8], [10]-[12].

When the similar parent recombination scheme is incorporated into NSGA-II, two parents with a small Hamming distance are recombined as shown in Fig. 1 (c). That is, the right-hand side (i.e., $H(P_1, P_2)$) of (6) is decreased by increasing the value of β in the similar parent recombination scheme. As a result, the two terms in the left-hand side of (6) are decreased as shown in Fig. 1 (d).

When the small gene exchange probability scheme is used, the right-hand side of (6) is not decreased as shown in Fig. 2 (c). However, one of the two terms in the

left-hand side of (6) is decreased as shown in Fig. 2 (d). That is, $\min\{H(C, P_1), H(C, P_2)\}$ becomes very small when the gene exchange probability α_u is very small.

Geometric crossover with the similar parent recombination scheme is illustrated in Fig. 3 (a) using two parents P_1, P_2 and its offspring C . In Fig. 3 (a), the horizontal and vertical axes show the Hamming distances from Parent P_1 and Parent P_2 , respectively. The short three arrows show the possible moves by mutation of a single gene of C . In Fig. 3 (a), $H(C, P_1) + H(C, P_2) = H(P_1, P_2)$ holds since $H(C, P_1) = 2, H(C, P_2) = 2$ and $H(P_1, P_2) = 4$. Similar parent recombination means that $H(P_1, P_2)$ is small in Fig. 3 (a).

Fig. 3 (b) illustrates geometric crossover with a small gene exchange probability. Since no mechanism for similar parent recombination is used, two parents have a larger Hamming distance in Fig. 3 (b) than Fig. 3 (a): $H(P_1, P_2) = 10$ in Fig. 3 (b). However, due to a small gene exchange probability, the generated offspring C is close to P_1 or P_2 as shown in Fig. 3 (b). In our computational experiments, one of the two offspring is randomly selected for further use in NSGA-II.

In uniform crossover with a very small gene exchange probability α_u , the expected value of $\min\{H(C, P_1), H(C, P_2)\}$ can be approximated by $\alpha_u H(P_1, P_2)$. For example, when $\alpha_u = 0.2$ and $H(P_1, P_2) = 10$, the expected value of $\min\{H(C, P_1), H(C, P_2)\}$ is 2. This means that the parent-offspring distance depends on the parent-parent distance, which depends on the number of objectives and the value of α_u as shown in Fig. 2 (c).

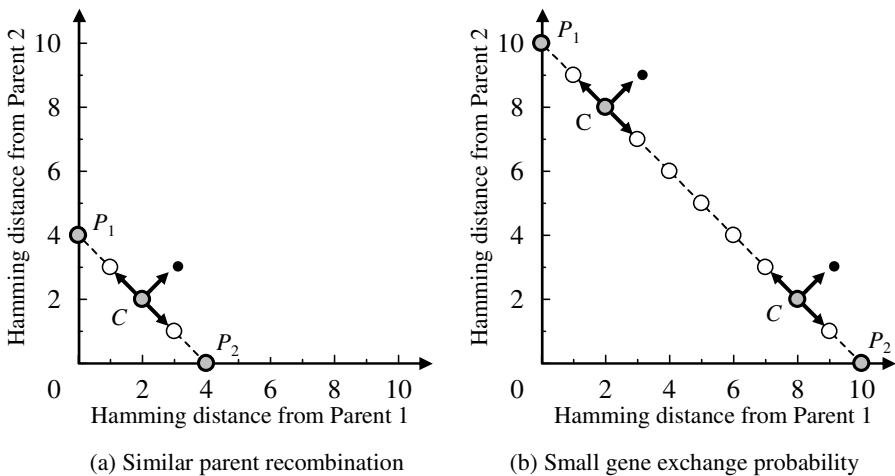


Fig. 3. Illustration of the two performance improvement schemes

In order to examine the relation between the parent-offspring distance and the performance of NSGA-II in a more straightforward manner, we implement the following distance-based crossover operator, which is incorporated into NSGA-II:

Distance-Based Crossover Operator:

1. The parent-offspring distance is specified as a user-defined parameter. This parameter is denoted by D .
2. Two parents P_1 and P_2 are selected in the same manner as in NSGA-II.

3. If $H(P_1, P_2) < 2D$, we cannot generate an offspring C such that $\min\{H(C, P_1), H(C, P_2)\} = D$ by geometric crossover (since $H(C, P_1) + H(C, P_2) < 2D$). In this case, we use standard uniform crossover for P_1 and P_2 . Otherwise (i.e., $H(P_1, P_2) \geq 2D$), D loci are randomly selected from $H(P_1, P_2)$ loci with different bit values in P_1 and P_2 . Each of those loci is selected uniformly with the same probability. The genes (i.e., different bit values) in the selected D loci are exchanged between P_1 and P_2 . Since the D genes are exchanged, $\min\{H(C, P_1), H(C, P_2)\} = D$ holds.
4. One of the generated two offspring is randomly selected for further use in NSGA-II (i.e., mutation is applied to the selected offspring in NSGA-II).

In the same manner as in Section 4, we apply NSGA-II with the distance-based crossover to our test problems. We examine the following parameter specifications: $D = 1, 2, 3, 4, 5, 10, 15, 20$. Experimental results are summarized in Fig. 4.

In Fig. 4 (a), the best results with respect to the hypervolume for the reference point $(0, 0, \dots, 0)$ are obtained when the parent-offspring distance is specified as 1 or 2. That is, the best results are obtained when different genes in one or two loci are exchanged between two parents. However, the distance-based crossover operator with such a parameter specification severely degrades the hypervolume for the reference point $(15000, 15000, \dots, 15000)$ in Fig. 4 (b). The hypervolume for this reference point for the 8-500 and 10-500 problems are clearly improved when the parent-offspring distance is specified as 10 or 15 in Fig. 4 (b). From Fig. 4 (c), we can see that 10 and 15 are much smaller than the average parent-parent distance.

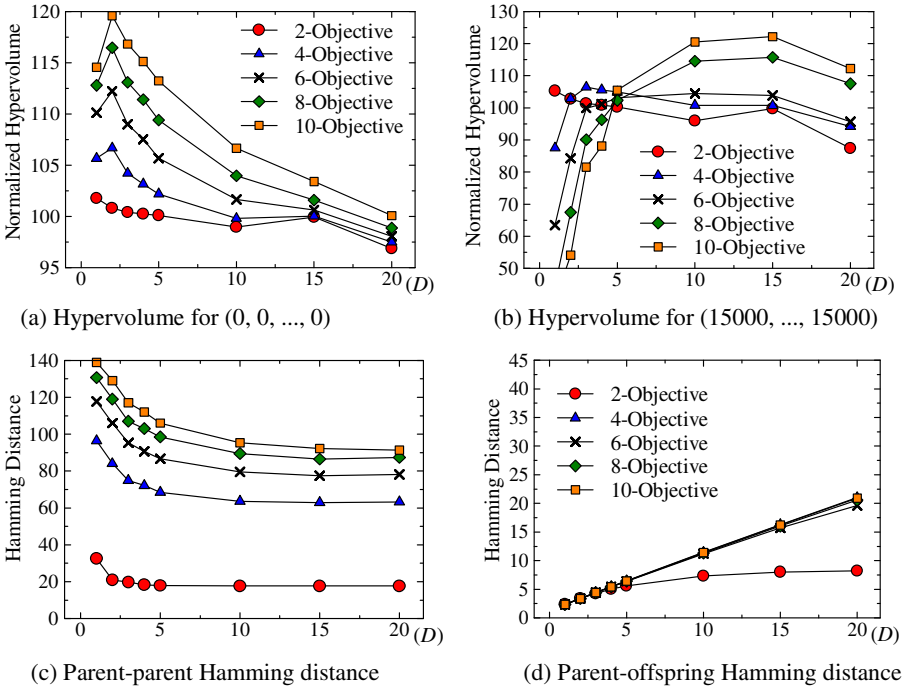


Fig. 4. Distance-based crossover with the parent-offspring distance D

One may notice that the average parent-offspring distance in Fig. 4 (d) for the 2-500 problem is much smaller than D when D is large. This is because the standard uniform crossover is used when $H(P_1, P_2) < 2D$ (i.e., see Fig. 4 (c) for $H(P_1, P_2)$).

Experimental results in Section 4 and Section 5 are summarized in Table 1 and Table 2 where the best average result by each scheme for each test problem is shown together with the standard deviation and the best parameter value in parentheses. The best result over the three schemes is highlighted by bold for each test problem. From Table 1 and Table 2, we can see that the similar parent recombination does not work well on the 6-500, 8-500 and 10-500 problems whereas it works well on the 2-500 and 4-500 problems. This is because a pair of similar parents is not actually selected (since all solutions in a population are not similar to each other in many-objective optimization). For example, in Fig. 1 (c), the average parent-parent distance is larger than 10 even when the most similar parent is selected from 50 candidates (i.e., $\beta = 50$) for the 10-500 problem. We can also see that the best specification of the parent-offspring distance D is surprisingly small in the last column of Table 1.

Table 1. Experimental results for the normalized average hypervolume for the reference point $(0, 0, \dots, 0)$. The best result by each scheme is shown for each test problem.

Problem	Similar Parent (SD) (β)	Probability (SD) (α_u)	Distance-Based (SD) (D)
2-500	102.3 (0.4) (20)	100.9 (0.4) (0.03)	101.6 (0.5) (1)
4-500	106.4 (0.9) (20)	104.8 (0.8) (0.02)	106.4 (0.8) (2)
6-500	108.5 (1.0) (20)	109.6 (1.0) (0.02)	112.1 (1.1) (2)
8-500	108.4 (1.5) (20)	113.4 (1.3) (0.02)	116.1 (1.2) (2)
10-500	108.2 (2.1) (20)	116.7 (1.6) (0.02)	119.7 (1.8) (2)

Table 2. Experimental results for the normalized average hypervolume for the reference point $(15000, 15000, \dots, 15000)$. The best result by each scheme is shown for each test problem.

Problem	Similar Parent (SD) (β)	Probability (SD) (α_u)	Distance-Based (SD) (D)
2-500	107.4 (1.2) (20)	102.8 (1.4) (0.03)	104.7 (1.5) (1)
4-500	112.9 (4.7) (10)	102.2 (5.1) (0.05)	106.1 (4.9) (3)
6-500	103.0 (14.9) (5)	107.2 (14.3) (0.10)	109.6 (12.5) (5)
8-500	99.5 (31.6) (5)	116.1 (21.9) (0.10)	115.7 (24.8) (15)
10-500	109.9 (50.9) (5)	124.3 (34.1) (0.15)	122.6 (36.6) (15)

6 Conclusions

In this paper, we examined the existing two schemes (i.e., the recombination of similar parents and the exchange of only a small number of genes) for improving the performance of EMO algorithm on multi-objective knapsack problems. For further discussing their effects, we also implemented a distance-based crossover where the distance from an offspring to its closer parent was specified in uniform crossover as a user-defined parameter. The performance of NSGA-II with each scheme was evaluated using the hypervolume values for two reference points. One is far from and the other is close to the Pareto front. For the reference point far from the Pareto front, good results were obtained for all test problems with 2-10 objectives when the

parent-offspring distance was very small (i.e., their Hamming distance was 1 for the 2-500 problem and 2 for the other problems). In this case, the diversity of solutions was very large. However, the convergence of solutions was severely deteriorated for many-objective problems. For the reference point close to the Pareto front, good results were obtained for many-objective knapsack problems with 8 and 10 objective when the parent-offspring distance was about 10-15.

References

1. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective Selection based on Dominated Hypervolume. *European J. of Operational Research* 181, 1653–1669 (2007)
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation* 6, 182–197 (2002)
3. Ishibuchi, H., Akedo, N., Nojima, Y.: Recombination of Similar Parents in SMS-EMOA on Many-Objective 0/1 Knapsack Problems. In: Coello Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 132–142. Springer, Heidelberg (2012)
4. Ishibuchi, H., Akedo, N., Nojima, Y.: Relation between Neighborhood Size and MOEA/D Performance on Many-Objective Problems. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 459–474. Springer, Heidelberg (2013)
5. Ishibuchi, H., Akedo, N., Nojima, Y.: Behavior of Multi-Objective Evolutionary Algorithms on Many-Objective Knapsack Problems. *IEEE Trans. on Evolutionary Computation* (in press)
6. Ishibuchi, H., Narukawa, K., Tsukamoto, N., Nojima, Y.: An Empirical Study on Similarity-Based Mating for Evolutionary Multiobjective Combinatorial Optimization. *European J. of Operational Research* 188, 57–75 (2008)
7. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary Many-Objective Optimization: A Short Review. In: Proc. of IEEE CEC, pp. 2424–2431 (2008)
8. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Diversity Improvement by Non-Geometric Binary Crossover in Evolutionary Multiobjective Optimization. *IEEE Trans. on Evolutionary Computation* 14, 985–998 (2010)
9. Jaszkiwicz, A.: On the Computational Efficiency of Multiple Objective Metaheuristics: The Knapsack Problem Case Study. *European J. of Operational Research* 158, 418–433 (2004)
10. Moraglio, A., Poli, R.: Topological Interpretation of Crossover. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1377–1388. Springer, Heidelberg (2004)
11. Moraglio, A., Poli, R.: Product Geometric Crossover. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN IX. LNCS, vol. 4193, pp. 1018–1027. Springer, Heidelberg (2006)
12. Moraglio, A., Poli, R.: Inbreeding Properties of Geometric Crossover and Non-geometric Recombinations. In: Stephens, C.R., Toussaint, M., Whitley, L.D., Stadler, P.F. (eds.) FOGA 2007. LNCS, vol. 4436, pp. 1–14. Springer, Heidelberg (2007)
13. Sato, H., Aguirre, H.E., Tanaka, K.: Local Dominance and Local Recombination in MOEAs on 0/1 Multiobjective Knapsack Problems. *European J. of Operational Research* 181, 1708–1723 (2007)

14. Sato, H., Aguirre, H., Tanaka, K.: Variable Space Diversity, Crossover and Mutation in MOEA Solving Many-Objective Knapsack Problems. *Annals of Mathematics and Artificial Intelligence* 68, 197–224 (2013)
15. While, L., Bradstreet, L., Barone, L.: A Fast Way of Calculating Exact Hypervolumes. *IEEE Trans. on Evolutionary Computation* 16, 86–95 (2012)
16. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. on Evolutionary Computation* 11, 712–731 (2007)
17. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Trans. on Evolutionary Computation* 3, 257–271 (1999)

Discovery of Implicit Objectives by Compression of Interaction Matrix in Test-Based Problems

Paweł Liskowski and Krzysztof Krawiec

Institute of Computing Science, Poznan University of Technology, Poznań, Poland
{pliskowski,krawiec}@cs.put.poznan.pl

Abstract. In test-based problems, commonly solved with competitive coevolution algorithms, candidate solutions (e.g., game strategies) are evaluated by interacting with tests (e.g., opponents). As the number of tests is typically large, it is expensive to calculate the exact value of objective function, and one has to elicit a useful training signal (search gradient) from the outcomes of a limited number of interactions between these coevolving entities. Averaging of interaction outcomes, typically used to that aim, ignores the fact that solutions often have to master different and unrelated *skills*, which form underlying objectives of the problem. We propose a method for on-line discovery of such objectives via heuristic compression of interaction outcomes. The compressed matrix implicitly defines derived search objectives that can be used by traditional multiobjective search techniques (NSGA-II in this study). When applied to the challenging variant of multi-choice Iterated Prisoner's Dilemma problem, the proposed approach outperforms conventional two-population coevolution in a statistically significant way.

Keywords: Test-based problems, coevolution, iterated prisoner dilemma, multiobjective evolutionary algorithms.

1 Introduction

Test-based problems are search and optimization tasks where candidate solutions are being evaluated by confronting them with *tests*. A single *interaction* between a candidate solution and a test produces a scalar outcome that reflects the capability of the former to *pass* the latter (expressed in the simplest case as a binary value). Canonical examples of test-based problems are games, where candidate solutions and tests are game strategies, while interactions boil down to playing games between them.

Solving a test-based problem consists in finding a candidate solution with certain properties. In the most common case, it should maximize the *expected utility*, i.e., the average outcome against all tests. Finding such a solution is challenging in many test-based problems, because the number of tests is usually large, and for some problems even infinite. This problem can be mitigated by estimating a solution's utility by confronting it with a *sample* of tests of a computationally manageable size. Some evolutionary algorithms (e.g., [3]) implement this idea by

maintaining a population of candidate solutions and assessing their fitness on a sample of tests generated at random. Competitive coevolution algorithms (e.g., [14]) rely on tests that dwell either in the same population (for one-population coevolution), or in a separate, coevolving population of tests (for two-population coevolution). Hybrid approaches of coevolution with random sampling have also been studied [12].

At first sight, averaging interaction outcomes over the available tests seems natural, as the fitness obtained in this way approximates the expected utility, i.e., the ultimate objective of the search process. If the test sample is drawn at random, that approximation is even unbiased. On the other hand, such aggregation inevitably incurs information loss. The outcomes of interactions may compensate each other, so that candidate solutions can receive the same fitness (and thus be indiscernible for selection), even if they fare very differently with particular tests. It becomes thus natural to ask: do we have to ‘compress’ all the information about interactions outcomes into one scalar value? Why not exploit it more carefully, for the sake of making search more efficient?

Several studies in the past have investigated the possibility of scrutinizing individual interaction outcomes and leveraging them for better performance of an evolutionary process. Bucci [2] and de Jong [6] introduced *coordinate systems* that compress the interaction outcomes into a multidimensional structure. Technically, given the dominance relation as defined by interaction outcomes (where every test is treated as a separate objective), a coordinate system can be constructed that preserves this relation while typically featuring a lower number of derived objectives (dimensions). Interestingly, every such objective can be said to express a certain *skill* exhibited by the candidate solutions.

Unfortunately, even with a moderately large number of tests, it is unlikely for any candidate solution to dominate (in the above sense) any other candidate solution in the population. From such a sparse dominance relation, it is hard to elicit any information that would efficiently drive the search process. In this paper we propose a heuristic method that compresses the original interaction outcomes into a few derived objectives in a ‘lossy’ manner. The method does not guarantee to preserve the original dominance structure, but always produces a low number of derived objectives that approximately capture the skills exhibited by the candidate solutions. The experiments with two-population coevolutionary algorithm demonstrate that the objectives obtained in this way can be better ‘search drivers’ than the conventional, averaging fitness function.

2 Background

Consider a test-based problem $(\mathbb{S}, \mathbb{T}, g)$, where \mathbb{S} is the set of all candidate solutions, \mathbb{T} is the set of all tests, and $g : \mathbb{S} \times \mathbb{T} \rightarrow [0, 1]$ is interaction function that characterizes solution’s capability to pass t . In particular, $g(s, t) = 1$ means that s passed test t and $g(s, t) = 0$ is interpreted as s failing t . Our goal is to solve this problem by finding a candidate solution that maximizes the expected utility, i.e., $s^* = \arg \max_{s \in \mathbb{S}} \sum_{t \in \mathbb{T}} g(s, t)$.

We approach the test-based problems using two-population coevolutionary algorithms, with a population of candidate solutions $S \subset \mathbb{S}$ of size m and a population of n tests $T \subset \mathbb{T}$. Evolving candidate solutions and tests in two separate populations has been shown superior to one-population configuration [13], as it allows them to specialize in their roles, where candidate solutions focus on maximizing the search objective, while tests are responsible for creating a *learning gradient* for them.

In the evaluation phase of evolutionary workflow, we apply g to $S \times T$ and obtain an $m \times n$ interaction matrix G , where rows correspond to candidate solutions and columns correspond to tests. G implicitly defines a dominance relation \succ between the elements of S : $s_i \succ s_k \iff \forall j g_{i,j} \geq g_{k,j} \wedge \exists j : g_{i,j} > g_{k,j}$, where $g_{i,j}$ denotes the outcome of interaction between the i th candidate solution and j th test, $i = 1, \dots, n, j = 1, \dots, m$. This dominance relation, built on a limited number of individuals in S and T , is transient and never reveals their full characteristics. We assume that G is the only information available at the given generation (S and T do not feature *archives*).

To perform selection of candidate solutions in S , i.e., to determine a subset $S' \subset S$ of ‘promising’ candidate solutions, a coevolutionary algorithm has to elicit information from G . Though there is a gamut of possible elicitation methods, we first consider the following two extremes.

1. The **direct approach** could consist in employing the original dominance relation \succ defined by G to carry out the selection process. Technically, one would assume that, for every test $t \in T$, the outcomes of interactions with t determines performance on the associated objective. In theory, this should enable applying conventional multiobjective evolutionary methods, like NSGA-II [7].

The advantage of the direct approach is that it relies on all available and undistorted information on interaction outcomes. The downside is that the likelihood of any solution in S dominating any other is low even for a moderate number of tests in T , and becomes even lower when the tests become diversified (which we want them to be). When the dominance relation becomes sparse, solutions are often incomparable, and there is no information to base the selection process on. Also, the conventional multiobjective evolutionary computation algorithms are known to perform well only if the number of objectives is moderate; while the population of tests in a typical coevolutionary setup hosts usually not a few, but at least a few dozens of tests.

2. **Scalarization.** As the other extreme, the information contained in G can be scalarized by aggregating the interaction outcomes over all tests available in T and adopting the resulting quantity as solutions’ fitness:

$$f(s) = \sum_{t \in T} g(s, t) \quad (1)$$

The fitness defined in this way can be subsequently used to run an ordinary selection stage (e.g., tournament selection). The advantage of this approach is that f (when normalized) is an estimator of expected utility, which is our external objective of the search process, so an algorithm’s ‘search driver’ is consistent with the ultimate search goal. On the downside, aggregation incurs information

loss, and many pairs of solutions that were originally incomparable can receive similar, if not identical, fitness (the latter case being particularly likely if the underlying interaction function assumes only a few values, which is common in test-based problems).

In this study, we are interested in combining the advantages of these approaches, i.e., to preserve *some* information on dominance while avoiding aggregating interaction outcomes into a single scalar value. To this aim, we will ‘compress’ the original information in G into a few *derived objectives*. In a similar spirit, several past studies on competitive coevolution [2,6,11] proposed formal methods for deriving coordinate systems (CS) from interaction matrices. However, they all implement an exact approach, i.e., the spatial arrangement of solutions in the CS exactly reproduces the original dominance structure. If that structure was sparse, such was also the structure of the derived CS (which typically manifested in the CS having many dimensions). As a result, CSs defined in this way are interesting tools for studying interaction outcomes, but not necessarily useful ‘search drivers’. The approach we propose in the next section, by compressing the interaction outcomes in a lossy manner, guarantees to result in a few, albeit inexact, derived objectives.

3 Objective Compression Algorithm

Given an $m \times n$ interaction matrix G , the algorithm proceeds in two stages:

1. Clustering of tests. We treat every column of G , i.e., a vector of interaction outcomes of all solutions from S with the specific test t , as a point in an m -dimensional space. A clustering algorithm of choice is applied to the n points obtained in this way. We employ k -means, as it is conceptually straightforward and typically converges in a few iterations. With k being the number of clusters, the outcome of this step is the clustering/partitioning $\{T_1, \dots, T_k\}$ of the original n tests (columns in G) into k subsets.

2. Defining derived objectives. For each cluster T_c , we derive from it a new search objective by averaging the corresponding columns in G row-wise. The resulting vector is the centroid of the cluster T_c . The overall outcome is an $m \times k$ *derived interaction matrix* G' , where the columns correspond to the new *derived objectives*, while the rows correspond to candidate solutions in S .

The resulting derived objectives can be subsequently used to guide the selection process, e.g., employed as objectives in the NSGA-II algorithm, as in the experimental part of this paper.

Example. Consider the matrix of interactions between the population of candidate solutions $S = \{a, b, c, d\}$ and the population of tests $T = \{t_1, t_2, t_3, t_4\}$, shown in Fig. 2a. Clearly, the only dominance holding in this space is $b \succ a$. The four-dimensional space of interaction outcomes is shown in two two-dimensional plots (Figs. 2b and 2c) that span $t_1 \times t_2$ and $t_3 \times t_4$, respectively. This helps to reveal that the performances of candidate solutions on the tests t_1 and t_3 are quite correlated. An analogous observation holds for t_2 and t_4 . Assume the

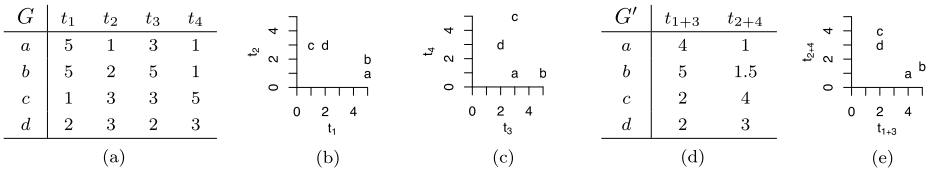


Fig. 1. Example of compression of interaction matrix (a) featuring a four-dimensional dominance structure (b, c), into a derived interaction matrix (d), resulting with the dominance structure shown in (e)

clustering algorithm decided to cluster t_1 with t_3 and t_2 with t_4 . The centroids of the clusters (Fig. 2d) form the derived objectives for this problem.

In the space of derived objectives (Fig. 2e) b still dominates a . However, now also c dominates d , though originally these two solutions were incomparable. As a result of compressing the original interaction matrix and merging dimensions, some information about the dominance structure has been lost.

In the particular case of c and d , introducing dominance in favor of c seems reasonable, as c outperforms d on two original objectives (t_3, t_4), while only one objective (t_1) supports the opposite relation (t_2 being neutral in this respect). However, the clustering of interaction matrix columns in general does not provide this kind of guarantees. In this sense, the above transformation trades the lower number of resulting objectives for certain inconsistency with the original interaction outcomes. Nevertheless, we posit that this imprecision may be a price worth paying for reducing the number of objectives and eliciting a potentially useful learning gradient from them.

Properties of the Method. The objective compression algorithm is allowed to distort the original dominance represented in the interaction matrix G . Though this can be considered a downside, note that the information in G does not fully characterize the candidate solutions in the first place, because of the limited number of tests available in T . In other words, it may not make sense to perfectly preserve the information in G if it is partial anyway (and not necessarily useful to drive the search process, as we discussed in Section 2). For the same reasons, we do not find it critical that the k -means algorithm employed here is a heuristic, and may produce different derived objectives depending on the initial random partitioning of original objectives.

The method features a few other properties. Trivially, clustering guarantees including any pair of original objectives that are mutually redundant (i.e., identical columns in G) into the same derived objective. Moreover, the more two tests are similar in terms of solutions’ performance on them, the more likely they will end up in the same cluster and contribute to the same derived objective. The clustering discovers thus certain *skills* of candidate solutions, as revealed in G .

For $k = 1$, the method degenerates to a single-objective approach (case #2 in Section 2): all tests form one cluster, and G' has a single column that contains solutions’ (normalized) estimate of expected utility defined in (1).

Setting $k = n$ implies $G' = G$, and the method implements the direct approach (case #1 in Section 2).

Finally, the derived objectives are additive components of scalar fitness (Eq. 1), i.e., $\sum_{j=1}^k g'_{i,j} = f(s_j)$, where the j th row in G' corresponds to s_j .

4 Experimental Verification

In the following we apply the proposed approach to the Iterated Prisoner's Dilemma (IPD), an abstract game that elegantly embodies the problem of achieving mutual cooperation in social, economic and biological interactions. The computational experiment is aimed at verification whether the objective compression algorithm influences the efficiency of coevolutionary learning.

Problem Definition. IPD is a two-player game involving a series of interactions, each of which is a Prisoner's Dilemma (PD) game. In a PD, a player can make one of two choices: cooperate or defect. If both players cooperate, they receive a payoff R , whereas if they both defect they get a smaller payoff P . Defecting against a cooperator gives a payoff T which is higher than R , and the cooperator in such a case receives the lowest possible payoff S . The PD payoff matrix must satisfy two conditions: $T > R > P > S$ and $2R > S + T$ [15].

In this study, we consider IPD extended to multiple choices (or *levels of cooperation*) [9,10,5,4] with payoff function that meets the above constraints:

$$p(c_A, c_B) = \begin{cases} 2.5 - 0.5c_A + 2c_B & \text{for player A} \\ 2.5 - 0.5c_B + 2c_A & \text{for player B} \end{cases}$$

An example of a payoff matrix for the 3-choice Prisoner's Dilemma generated using the above function is shown in Table 1, the possible choices being $\{-1, 0, 1\}$.

Strategy Representation. We adopt the direct look-up table [1] to represent the strategies (candidate solutions and tests) and consider the *memory-one* form of IPD in which the players remember their moves from the previous iteration only. In such a case, the n -choice IPD strategy is an $n \times n$ matrix M , where m_{ij} for $i, j = 1, 2, \dots, n$ specifies the choice to be made given the player's previous move i and the opponent's previous move j . The other element of the strategy is the initial move m_{00} .

In the experiments, we focus on IPD with $n = 9$ choices, which we found to be much more demanding than 3-choice IPD used in some earlier coevolutionary investigations [3]. Each strategy is a look-up table containing $9 \times 9 + 1 = 82$ choices and the size of search space is $9^{82} \approx 1.77 \times 10^{78}$.

Experimental Setup. We embed the objective compression algorithm in a typical two-population competitive coevolutionary setting (Section 2). The resulting $m \times k$ matrix of derived objectives forms the input for the NSGA-II algorithm [7] where they guide the selection process of candidate solutions. The role of NSGA-II is to maintain a diverse front of well-performing candidate solutions

Table 1. An exemplary payoff matrix for the 3-choice prisoner’s dilemma. Choice -1 is full defection, 1 is full cooperation. (p_A, p_B) denotes payoffs to players A and B , respectively.

		Player B		
		choice	1	0
Player A	1	(4, 4)	(2, 4.5)	(0, 5)
	0	(4.5, 2)	(2.5, 2.5)	(0.5, 3)
	-1	(5, 0)	(3, 0.5)	(1, 1)

by Pareto-ranking the population and resolving ties on selection by means of crowding distance.

In the population of tests, individuals are rewarded for making *distinctions* [8] between candidate solutions. This fitness function promotes tests that differentiate the candidate solutions and thus provide *search gradient* for them.

The objective compression algorithm (k -MEANS in the following) is examined in the presence of two control setups. In the first one, we replace the k -means algorithm with a naive approach to clustering in which individuals are assigned to clusters randomly (k -RAND). This configuration is intended to control for the relevance of clusters discovered by clustering. The second setup is an ordinary two-population coevolutionary algorithm (CEL, [14]), which is equivalent to 1-MEANS (see the discussion of algorithm properties at the end of Section 3).

All algorithms maintain populations of 50 candidate solutions and 50 tests. However, because NSGA-II effectively merges parents and offspring prior to selection (and in this sense features an internal archive), we set the candidate solutions population size to 100 for CEL. This provides for fair comparison between the methods. As a result, in each generation of every method, $100 \times 50 = 5,000$ IPD games are played, each of which consists of 150 PD episodes. Since each run consists of 200 generations, it requires the total effort of 1,000,000 games.

Both populations use tournament selection with tournament size 5. The only source of genetic variation is simple mutation which iterates over all elements of the look-up table and with probability 0.2 replaces the original choice with one of the remaining choices, selected at random. This operator has been found to provide sufficient variation of behaviors for multiple-choice IPD [4].

Results. Algorithms that solve test-based problems do not rely on an objective performance measure, so a candidate solution deemed good by an algorithm does not have to be such in reality. In other words, fitness as defined in an algorithm (if any) is *subjective* (internal) and may strongly differ from the true performance of a candidate solution. To *objectively* (externally) assess the performance of a candidate solution, we estimate its expected utility by letting it play 50,000 games against random opponents. Every random opponent is obtained independently by filling the look-up table with random choices. This assessment, commonly used with test-based problems, is external to a search algorithm and does not affect its behavior. Below we report algorithm performance meant in this way.

We performed separate experiments for $k = 2 \dots 5$ clusters. Let us first discuss the search dynamics, illustrated in Fig. 2a, which reports the objective performance of the best-of-generation candidate solutions, averaged over 120

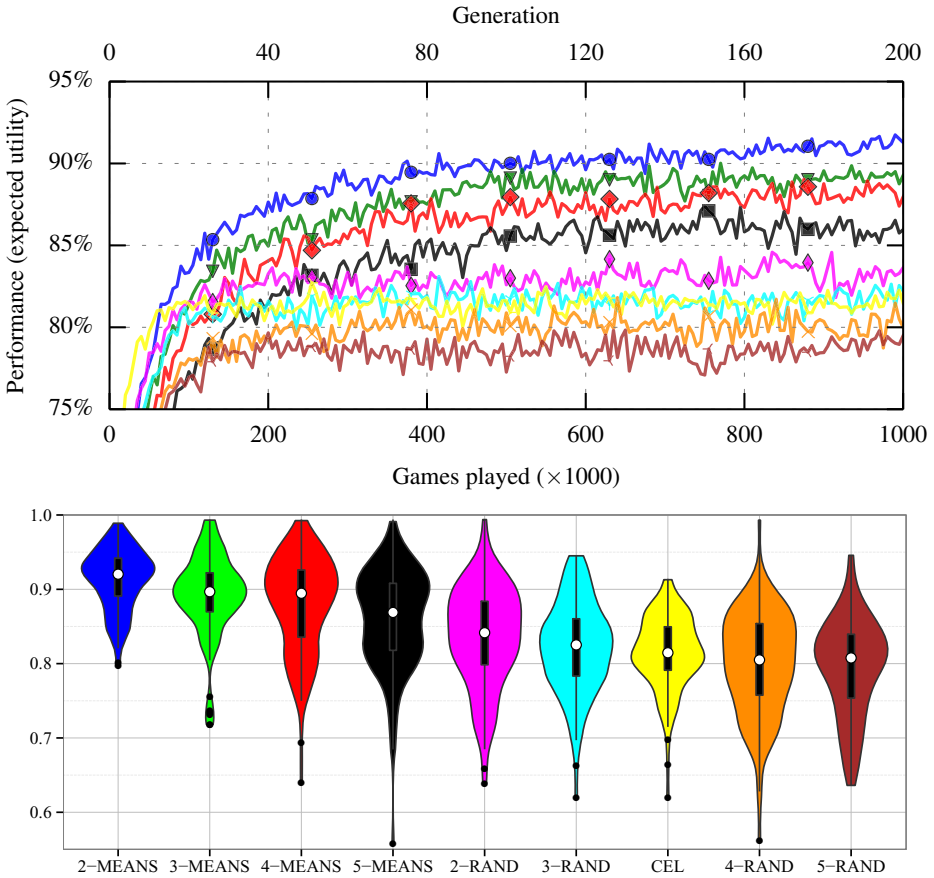


Fig. 2. The expected utility of best-of-generation individuals averaged over 120 runs (top) and distributions of expected utilities of the best-of-runs individuals (bottom). Violin plot legend: white dot: median, black box: interquartile range, line: 1.5 interquartile range, black dots: outliers. Both plots use the same colors.

coevolutionary runs. For reference, we plot the best-of-generation players found by standard coevolutionary search (CEL). Clearly, compression of the interaction outcomes allows k -MEANS outperform CEL for every k . In particular, $k = 2$ seems to be optimal, suggesting that this may be the required number of skills to effectively play the 9-choice IPD. We speculate that the two discovered skills correspond to full defection and full cooperation in IPD, since defecting is the dominant strategy of the game, while mutual cooperation pays off the most in the long term. This hypothesis requires, however, verification in the future.

Interestingly, we also observe certain positive influence of decomposing the scalar fitness function by *random* clustering (RAND). In this case however, improvement over CEL is noticeable only for $k = 2$ (2-RAND). Apparently, merging randomly selected tests into derived objectives cannot efficiently capture meaningful

Table 2. Expected utilities and 95% confidence intervals of best-of-run individuals obtained by the algorithms for 9-choice Iterated Prisoner’s Dilemma

Method	Expected utility	Method	Expected utility
CEL	81.48 ± 0.90	2-RAND	83.64 ± 1.21
2-MEANS	91.28 ± 0.73	3-RAND	82.21 ± 1.11
3-MEANS	89.27 ± 0.99	4-RAND	79.99 ± 1.23
4-MEANS	87.96 ± 1.16	5-RAND	79.49 ± 1.17
5-MEANS	85.96 ± 1.20		

skills that would effectively guide the multiobjective learning process towards improving the population of candidate solutions.

When it comes to comparing algorithms’ end-of-run outcome, Table 2 presents the average performance of the best-of-run individuals for each algorithm, accompanied by 95% confidence intervals, while Fig. 2b the distributions as violin plots. To compare these final performances of the algorithms, we applied Shapiro-Wilk test, which indicated the performance distributions to be likely non-normal (all p-values < 10^{-6}). We then conducted the nonparametric Kruskal-Wallis rank sum test, which revealed a statistically significant ($\chi^2 = 158.7$, p-value < 2.2×10^{-16}) difference between the results obtained by particular algorithms. A post-hoc analysis using pairwise Wilcoxon rank sum test with Holm correction indicated the following ranking among the configurations:

$$2\text{-MEANS} > 3\text{-MEANS} > 4\text{-MEANS} = 5\text{-MEANS} > 2\text{-RAND} = 3\text{-RAND} = 4\text{-RAND} = 5\text{-RAND} = \text{CEL}$$

where ‘>’ denotes significant difference and ‘=’ means no statistical difference. This ranking corroborates our earlier observations: meaningful grouping of tests (and associated original objectives) and using the resulting derived objectives in multiobjective setting makes coevolutionary search more effective.

5 Conclusions

In this paper we proposed a heuristic method that compresses the original interaction outcomes into a few derived, implicit objectives in a lossy manner. Even though the method does not guarantee to preserve the original dominance structure, it successfully manages to produce a low number of objectives that approximately capture the skills exhibited by the candidate solutions. Crucially, our method avoids aggregating interaction outcomes into a scalar value, therefore allowing multiobjective approach to the problem. We demonstrated how this compression, combined with the NSGA-II algorithm, can be applied to effectively enhance the coevolutionary search.

The experiments with two-population coevolutionary algorithm demonstrate that the implicit objectives discovered in interaction outcomes are indeed better search drivers than the conventional, averaging fitness function. Our results support the claim that it is enough to preserve only some information on dominance to obtain a useful learning gradient. Applicability of this approach to other interactive and non-interactive domains is to be verified in future research.

Acknowledgments. P. Liskowski acknowledges support from grant no. DEC-2013/09/D/ST6/03932 and K. Krawiec acknowledges support from grant no. DEC-2011/01/B/ST6/07318.

References

1. Axelrod, R.: The evolution of strategies in the iterated prisoner's dilemma. *The Dynamics of Norms*, 1–16 (1987)
2. Bucci, A., Pollack, J.B., de Jong, E.: Automated extraction of problem structure. In: Deb, K., Tari, Z. (eds.) *GECCO 2004, Part I. LNCS*, vol. 3102, pp. 501–512. Springer, Heidelberg (2004)
3. Chong, S.Y., Tino, P., Ku, D.C., Xin, Y.: Improving Generalization Performance in Co-Evolutionary Learning. *IEEE Transactions on Evolutionary Computation* 16(1), 70–85 (2012)
4. Chong, S.Y., Yao, X.: Behavioral diversity, choices and noise in the iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation* 9(6), 540–551 (2005)
5. Darwen, P.J., Yao, X.: Why more choices cause less cooperation in iterated prisoner's dilemma. In: *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 2, pp. 987–994. IEEE (2001)
6. de Jong, E.D., Bucci, A.: DECA: Dimension extracting coevolutionary algorithm. In: Cattolico, M.C., et al. (eds.) *GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, Washington, USA, pp. 313–320. ACM Press (2006)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
8. Ficici, S.G., Pollack, J.B.: Pareto optimality in coevolutionary learning. In: Kelemen, J., Sosík, P. (eds.) *ECAL 2001. LNCS (LNAI)*, vol. 2159, pp. 316–325. Springer, Heidelberg (2001)
9. Frean, M.: The evolution of degrees of cooperation. *Journal of Theoretical Biology* 182(4), 549–559 (1996)
10. Harrald, P.G., Fogel, D.B.: Evolving continuous behaviors in the iterated prisoner's dilemma. *Biosystems* 37(1), 135–145 (1996)
11. Jaśkowski, W., Krawiec, K.: Formal analysis, hardness, and algorithms for extracting internal structure of test-based problems. *Evolutionary Computation* 19(4), 639–671 (2011)
12. Jaśkowski, W., Liskowski, P., Szubert, M., Krawiec, K.: Improving coevolution by random sampling. In: Blum, C. (ed.) *GECCO 2013: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, Amsterdam, The Netherlands, pp. 1141–1148. ACM (2013)
13. Juillé, H., Pollack, J.B.: Coevolving the "ideal" trainer: Application to the discovery of cellular automata rules. University of Wisconsin, pp. 519–527. Morgan Kaufmann (1998)
14. Popovici, E., Bucci, A., Wiegand, R.P., de Jong, E.D.: *Coevolutionary Principles*. In: *Handbook of Natural Computing*. Springer (2011)
15. Poundstone, W.: *Prisoner's Dilemma: John von Neuman, Game Theory, and the Puzzle of the Bomb*. Doubleday, New York (1992)

Local Optimal Sets and Bounded Archiving on Multi-objective NK-Landscapes with Correlated Objectives

Manuel López-Ibáñez¹, Arnaud Liefooghe², and Sébastien Verel³

¹ IRIDIA, Université Libre de Bruxelles (ULB), Brussels, Belgium
`manuel.lopez-ibanez@ulb.ac.be`

² Université Lille 1, LIFL, UMR CNRS 8022, Inria Lille-Nord Europe, France
`arnaud.liefooghe@univ-lille1.fr`

³ Université du Littoral Côte d'Opale, LISIC, France
`verel@lisic.univ-littoral.fr`

Abstract. The properties of local optimal solutions in multi-objective combinatorial optimization problems are crucial for the effectiveness of local search algorithms, particularly when these algorithms are based on Pareto dominance. Such local search algorithms typically return a set of mutually nondominated Pareto local optimal (PLO) solutions, that is, a PLO-set. This paper investigates two aspects of PLO-sets by means of experiments with Pareto local search (PLS). First, we examine the impact of several problem characteristics on the properties of PLO-sets for multi-objective NK-landscapes with correlated objectives. In particular, we report that either increasing the number of objectives or decreasing the correlation between objectives leads to an exponential increment on the size of PLO-sets, whereas the variable correlation has only a minor effect. Second, we study the running time and the quality reached when using bounding archiving methods to limit the size of the archive handled by PLS, and thus, the maximum size of the PLO-set found. We argue that there is a clear relationship between the running time of PLS and the difficulty of a problem instance.

1 Introduction

Several state-of-the-art algorithms for multi-objective combinatorial optimization problems (MCOPs) are based on local search. These local search algorithms are either based on solving multiple scalarizations of the objective function vector, or they are based on Pareto dominance. The most successful local search algorithms combine both approaches [3,4,10,13]. These successes explain the increasing interest on understanding the role of local optimal solutions in the context of MCOPs. Previous works have focused on the properties of individual local optimal solutions [14]. However, algorithms for MCOPs typically return not a single solution, but a set of solutions that approximates the Pareto set. Thus, local search algorithms for MCOPs are typically concerned by (Pareto) local optimal *sets* (PLO-sets), that is, sets of (Pareto) local optimal solutions where

solutions are mutually nondominated and are also local optimal with respect to the neighborhood of the other solutions in the set [12].

Experimental studies on PLO-sets have been so far limited to the study of some of their properties for the bi-objective traveling salesman problem, in particular the number of solutions in each PLO-set and the connectedness of PLO-sets [11]. This paper extends significantly this initial work in two aspects. First, we consider multi-objective NK-landscapes with correlated objectives (ρ MNK-landscapes) [14], which allow us to examine the effect of various problem characteristics on the properties of PLO-sets. Second, we examine the effect of using bounded archiving methods [7,9] in order to limit the size of the PLO-set handled by the local search. Archiving methods are often used in both local search and evolutionary multi-objective algorithms, and there are a few recent works on their theoretical properties [2,8,9]. Several authors have mentioned as interesting future work the experimental study of the PLO-sets induced by bounded archiving. To the best of our knowledge, we present in this paper the first results of such a study. Our conclusions not only give support to previous theoretical results, but also give insights on how to improve local search algorithms for particularly difficult MCOPs. In the following, we recall the required background on problems and algorithms in Section 2; we provide the experimental setup of our study in Section 3; we discuss our experimental results in Section 4; and we conclude in the last section.

2 Background

Multi-Objective Combinatorial Optimization. A *multi-objective combinatorial optimization problem* (MCOP) is defined by an objective function vector $f = (f_1, \dots, f_m)$ with $m \geq 2$ objective functions, and a discrete set X of feasible solutions in the *solution space*. Let $Z = f(X) \subseteq \mathbb{R}^m$ be the set of feasible outcome vectors in the *objective space*. Each solution $x \in X$ is assigned an objective vector $z \in Z$ on the basis of the multidimensional function vector $f: X \rightarrow Z$ such that $z = f(x)$. When all objectives are to be maximized, a solution x weakly dominates another solution x' if $\forall i \in \{1, \dots, m\}, f_i(x) \geq f_i(x')$. If, in addition, $\exists i \in \{1, \dots, m\}$ such that $f_i(x) > f_i(x')$, then we say that x dominates x' ($x \prec x'$). When $x \not\prec x' \wedge x' \not\prec x$, we say that x and x' are mutually nondominated. A solution $x^* \in X$ is *Pareto optimal* if $\nexists x \in X$ such that $x \prec x^*$. The set of all Pareto optimal solutions is the *Pareto set*. Its mapping in the objective space is the *Pareto front*. One of the goals in multi-objective optimization is to identify the Pareto set, or a good approximation of it.

Pareto Local Optimal Sets. Set-based local search algorithms for MCOPs generally combine the use of a neighborhood operator with the management of an archive of mutually nondominated solutions found so far. A *neighborhood operator* is a mapping function $\mathcal{N}: X \rightarrow 2^X$ that assigns a set of solutions $\mathcal{N}(x) \subset X$ to any solution $x \in X$. $\mathcal{N}(x)$ is called the *neighborhood* of x , and a solution $x' \in \mathcal{N}(x)$ is called a *neighbor* of x . A solution $x \in X$ is a *Pareto local optimum* (PLO) with respect to a neighborhood structure \mathcal{N} if there is no

neighbor $x' \in \mathcal{N}(x)$ such that $x' \prec x$ [12]. A set $S \subseteq X$ is a *Pareto local optimal set (PLO-set)* with respect to \mathcal{N} if, and only if, it contains only PLO-solutions with respect to \mathcal{N} and all solutions are mutually nondominated [12]. In addition, a set $S \subseteq X$ is a *maximal PLO-set* with respect to \mathcal{N} if, and only if, $\forall s' \in \mathcal{N}(S)$, $\exists s \in S$ such that $s \prec s' \vee f(s) = f(s')$, where $\mathcal{N}(S) = \bigcup_{s \in S} \mathcal{N}(s)$ [12]. In other words, any neighbor of any solution in a maximal PLO-set is weakly dominated by a solution in the set.

Pareto Local Search. A typical example of a multi-objective local search algorithm is Pareto Local Search (PLS) [12]. PLS is an extension of the conventional hill-climbing algorithm to the multi-objective case. An archive of nondominated solutions is initialized with at least one solution. At each iteration, one solution is chosen at random from the archive and all its neighbors are evaluated and compared against the archive. Each neighbor is added to the archive, and marked as *unvisited*, if it is not dominated by any other solution in the archive. Moreover, solutions in the archive dominated by this neighbor are removed. Once all neighbors have been evaluated, the current solution is marked as *visited*. The algorithm stops once all solutions from the archive are marked as *visited*.

PLS is known to be a well-performing algorithm, either as a stand-alone approach or as a hybrid component, for many MCOPs [3,4,10,13]. Moreover, independently of the initial archive, PLS always terminates and returns a maximal PLO-set [12]. However, the archive of (unvisited) solutions may grow exponentially with respect to the instance size and, in that case, PLS may require an exponential number of iterations. In such a situation, it would be more interesting to bound the size of the archive in order to prevent an exponential grow, but still return a (perhaps non-maximal) PLO-set.

Given an archive A and a maximum size μ , a bounded archiving method, or *archiver* for short, will return a new archive $A' \subseteq A$ such that $|A'| \leq \mu$ [7,9]. We can use an archiving method in PLS such that whenever a new solution x' is added to the archive A and $|A \cup \{x'\}| = \mu + 1$, then the archiving method will select one solution to be removed. This is equivalent to the $\mu + \lambda$ strategy [2], with $\lambda = 1$. The various archiving methods differ on how the solution to be removed is selected. Here we focus on two archiving methods, hypervolume archiver (HVA) [6] and multi-level grid archiver (MGA) [8], which are the only known archiving methods belonging to the class with the most desirable convergence properties [9]. Two of these properties are: (i) accepting solutions outside the objective space region dominating the current archive (*diversifies*) and (ii) a subsequent archive cannot be worse in terms of Pareto dominance than an earlier archive (\prec -*monotone*).

When PLS uses either HVA or MGA as its archiving method, then a run of PLS will stop at a PLO-set, but not necessarily at a maximal PLO-set. Indeed, there may exist a solution $s' \in \mathcal{N}(A)$, such that $\nexists s \in A$ with $s \prec s'$, but the archiving method chose to discard s' in order to maintain $|A| \leq \mu$. Therefore, it is expected that when using an archiving method, PLS will converge faster but to a possibly worse PLO-set. Moreover, since the decision of which solutions are discarded are fundamentally different for HVA and MGA, we would expect

that each method may converge to disjoint sets of PLO-sets. In the experiments presented here, we study the properties of the PLO-sets returned by the classical PLS (using an unbounded archive) and when PLS uses either HVA or MGA to bound the archive size.

Multi-Objective NK-landscapes with Correlated Objectives. We study the effect of various characteristics of MCOPs on the properties of PLO-sets by means of ρ MNK-landscapes, which are artificial multi-objective multimodal problems with objective correlation [14]. They extend both single-objective NK-landscapes [5] and multi-objective NK-landscapes with independent objective functions [1]. Feasible solutions are binary strings of size n . The parameter k refers to the number of variables that influence a particular position from the bit-string (the epistatic interactions). The objective function vector is defined as $f: \{0, 1\}^n \rightarrow [0, 1]^m$. Each objective function is to be maximized, and can be formalized as follows: $f_i(x) = \frac{1}{n} \sum_{j=1}^n c_j^i(x_j, x_{j_1}, \dots, x_{j_k})$, $i \in \{1, \dots, m\}$, where $c_j^i: \{0, 1\}^{k+1} \rightarrow [0, 1]$ defines the component function associated with each variable x_j , $j \in \{1, \dots, n\}$, for objective f_i , and where $k < n$. By increasing the number of variable interactions k from 0 to $(n - 1)$, ρ MNK-landscapes can be gradually tuned from smooth to rugged.

We generate an instance of a ρ MNK-landscape by randomly setting the position of these epistatic interactions, following a uniform distribution. The same epistatic degree k and the same epistatic interactions are used for all the objectives. Component function values are sampled within the range $[0, 1)$ following a multivariate uniform distribution of dimension m with a correlation coefficient ρ . A positive (resp. negative) correlation coefficient decreases (resp. increases) the degree of conflict between the objective function values.

3 Experimental Setup

In the following, we investigate ρ MNK-landscapes with a problem size $n \in \{8, 16\}$, an epistatic degree $k \in \{1, 2, 4, 8\}$ such that $k < n$, an objective space dimension $m \in \{2, 3, 5\}$, and an objective correlation $\rho \in \{-0.7, -0.2, 0.0, 0.2, 0.7\}$ such that $\rho > \frac{-1}{m-1}$, because the corresponding correlation matrix is symmetric positive-definite [14]. The investigated problem sizes allow us to enumerate the solution space exhaustively, and then to solve all the instances to optimality. One independent random instance is considered for each parameter combination: $\langle \rho, m, n, k \rangle$. This leads to a total of 91 problem instances.

In our implementation of PLS for ρ MNK-landscapes, the neighborhood structure is taken as the *1-bit-flip*. The archive is initialized with one random solution from the solution space. At each iteration, the neighborhood of the selected solution is explored exhaustively in a random order. This order has an impact on the dynamics of PLS since bounded archiving methods treat incoming solutions sequentially. The cost of each iteration of PLS is exactly n evaluations, corresponding to the neighborhood size. We experiment with three variants of PLS. PLS_{UNB} corresponds to the classical PLS with an unbounded archive.

PLS_{HVA} and PLS_{MGA} use HVA and MGA, respectively, to bound the size of the archive to a maximum of μ solutions, where $\mu \in \{10, 20, 40, 80\}$. We consider 25 different seeds for the random number generator used in PLS, and we run each PLS variant on each problem instance using each random seed. This leads to a total of 20 475 runs.

4 Experimental Analysis

4.1 Cardinality of Pareto Local Optimal Sets

Figure 1 shows the size of the PLO-sets identified by PLS_{UNB} with respect to different instance characteristics. Each point gives the mean value over the 25 random seeds on the same instance, and the error bars indicate the standard deviation. PLO-set sizes are plotted in logarithmic scale.

In general, the cardinality of PLO-sets increases exponentially with the number of objectives m , as shown in Figs. 1a–1c. The correlation between objective values ρ is also a crucial factor: the cardinality of the PLO-sets increases exponentially with the linear decrease of ρ (Fig 1c). The cardinality of PLO-sets also increases with lower k , but much less noticeably (Fig. 1a). Moreover, the small error bars in Figs. 1a–1d indicate that, for a given instance, maximal PLO-sets consistently have roughly the same size.

Given the above results, restricting the size of the PLO-sets by using bounding archiving methods must have a stronger effect as the correlation decreases and the number of objectives increases. In the next section, we examine this effect for each archiving method.

4.2 Quality of Local Optimal Sets

We examine the quality of the PLO-sets found by PLS_{UNB} , PLS_{HVA} and PLS_{MGA} in terms of two unary quality measures, namely, the hypervolume and the multiplicative epsilon [15]. In order to compare the hypervolume value for instances with very different characteristics, we compute the hypervolume relative difference as $hvr(A) = (hv(P) - hv(A))/hv(P)$, where $A \subseteq Z$ is the image of a PLO-set in the objective space and P is the exact Pareto front for the instance under consideration. The reference point is set to the origin. The epsilon measure gives the minimum multiplicative factor by which a PLO-set has to be shifted in the objective space to weakly dominate the exact Pareto front. Thus, for both measures, a lower value is preferred. Results are reported in Fig. 2 for different parameter settings.

As conjectured above, a first observation is that, as the size of maximal PLO-sets increases, the quality of PLO-sets obtained by bounded archiving decreases. The increase in quality is almost logarithmic with respect to the archive size limit (μ), as shown in Figs. 2a–2d. Here we show only results for $m = 5$, but the trends are similar for a lower number of objectives, although less pronounced. This trend appears independently of whether we measure quality in terms of

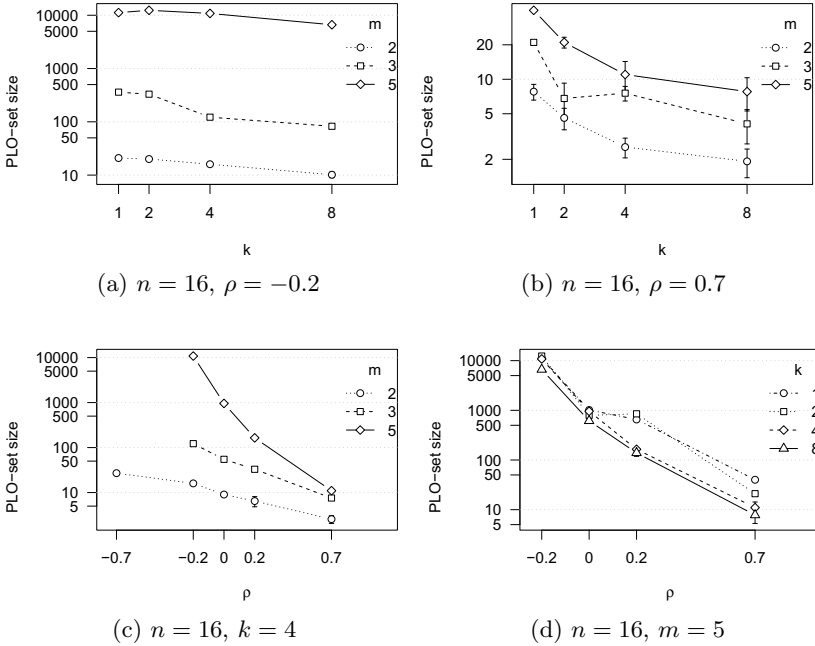


Fig. 1. Mean size of the PLO-sets returned by PLS_{UNB} . Error bars give the standard deviation.

epsilon or hypervolume. Moreover, as in the single-objective case, the average quality of local optima decreases with k [5], as shown in Figs. 2e–2f.

Lastly, there is not much difference in terms of quality between PLS_{HVA} and PLS_{MGA} . When differences occur, the PLO-sets returned by PLS_{HVA} have a better hypervolume (lower hvr value) than those returned by PLS_{MGA} (Figs. 2c–2d), however, there is no clear winner in terms of epsilon value (Fig. 2a–2b).

4.3 Difficulty of Identifying Local Optimal Sets

For single-objective NK-landscapes, the number of iterations, or steps, of a conventional hill-climbing algorithm provides an estimation of the average diameter of the basins of attraction of local optima [5]. This diameter characterizes a problem instance in terms of multimodality: The larger the length, the larger the basin diameter and the lower the number of local optima. Conversely, the smaller the length, the smaller the basin diameter and the higher the number of local optima. Multimodality characterizes an important aspect of instance difficulty (i.e., the number of local optima).

As in the single-objective case, the construction of ρMNK -landscapes implies that they are isotropic, that the neighborhood has the same properties in every direction of the objective space, and that the basins of attraction have a ball-like shape. In this section, we report the length of PLS for different archiving

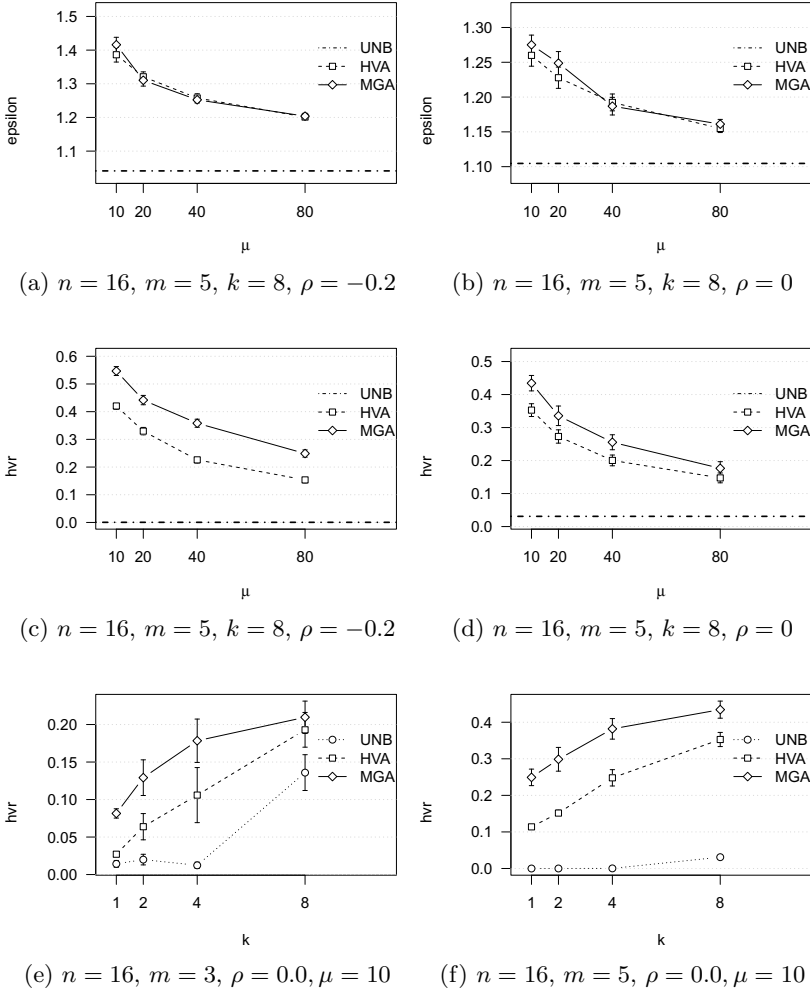


Fig. 2. Mean quality of the PLO-sets found by PLS_{UNB} , PLS_{HVA} and PLS_{MGA} measured in terms of multiplicative epsilon (*epsilon*) and hypervolume relative difference (*hvr*). Error bars give the standard deviation.

strategies. This allows us to study the running time, in terms of number of iterations, required by PLS to identify a PLO-set. Moreover, when the PLS length is smaller under one setting than another, we can reasonably assume that there exist more PLO-sets in the corresponding landscape since the search process got stuck more easily on a local optima. Figure 3 shows the PLS length for the same problem instances shown in Fig. 2. The comparison between each pair of plots shows clearly a relationship between the length of PLS (Fig. 3) and the quality of the results (Fig. 2): Larger length corresponds to better quality in general. Interestingly, bounding the archive size substantially reduces both the

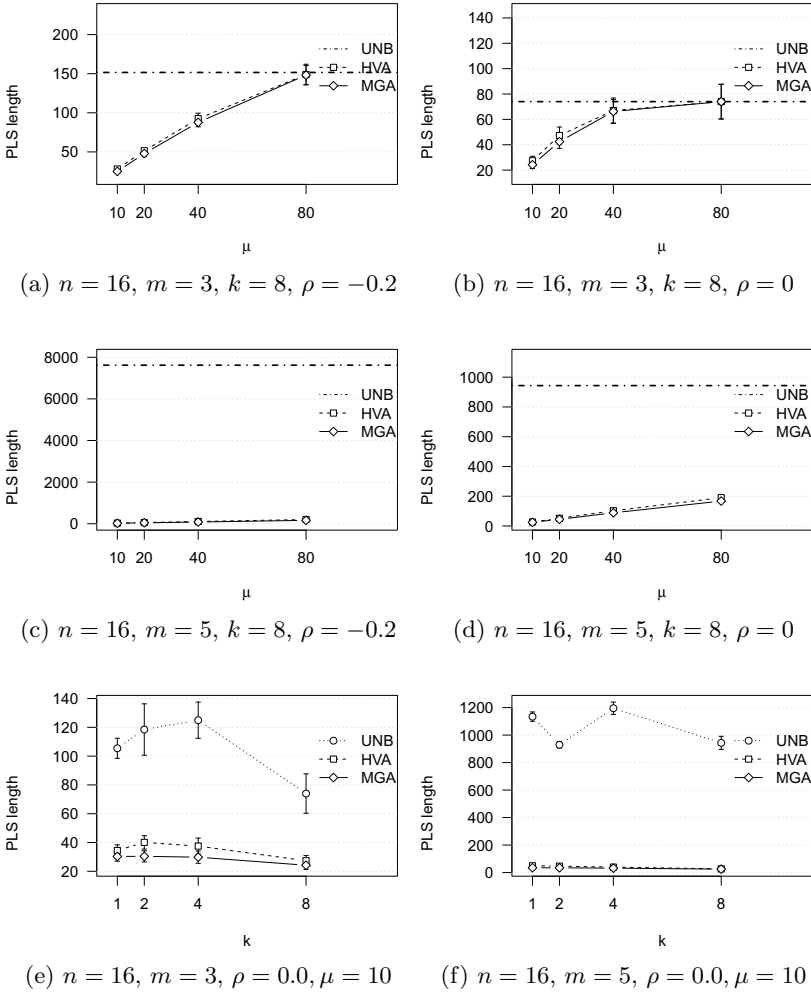


Fig. 3. Mean length of PLS_{UNB} , PLS_{HVA} and PLS_{MGA} . Error bars give the standard deviation.

quality of the obtained approximation as well as the running time of PLS. The smaller the archive size, the larger the difference with PLS_{UNB} . However, this also has the effect of increasing the number of PLO-sets. Indeed, when the archive size is small, the PLS length is small, which suggests that the average distance between a local optimum and the solutions from the corresponding basin of attraction is also small and, hence, the number of PLO-sets is large.

Surprisingly, when the archive is unbounded as reported in Fig. 1, the PLS length increases from $k = 1$ to $k = 4$ (despite we know that the number of PLO-solutions increases linearly with k [14]) and only becomes smaller for $k = 8$ (Fig. 3). This increase in PLS length contradicts known results from single-objective

optimization [5]. In fact, in the case of PLO-sets, both the number of PLO-solutions and the average size of the neighborhood of a PLO-set influence the number of PLO-sets. First, when the number of PLO-solutions increases, it is expected that the number of PLO-sets also increases, which will decrease the PLS length. By contrast, when a PLO-set is larger, the number of neighbor solutions in this set is larger as well. This potentially reduces the number of PLO-sets, making PLS run longer. The balance between these two effects could explain the PLS length; i.e., the PLS length starts to decrease when the number of PLO-solutions has a larger impact on the number of PLO-sets than the size of PLO-sets. Overall, these results suggest that the length of PLS could provide an estimation of the number of PLO-sets, and thus a measure of difficulty for archive-based local search.

5 Conclusions

In this paper, we analyzed the characteristics of local optima in set-based multi-objective local search when applied to multi-objective NK-landscapes with correlated objectives. First, the main factors affecting the cardinality of the maximal PLO-sets returned by PLS_{UNB} are the number of objectives and the correlation between them. By changing these two factors, the PLO-set size can vary from a few tens to tens of thousands. Our results confirm trends already noticed for other MCOP problems. In particular, the exponential increase in the PLO-set size with lower objective correlation has already been reported for the bi-objective QAP [13]. Another interesting observation is that, given a particular instance, the variability of PLO-set sizes is usually a very small fraction of the average size, that is, most maximal PLO-sets for a given instance have roughly the same size. This is not an obvious conjecture to make, and we currently do not know if it is also the case for other MCOPs. Our experiments also strongly indicate that the relationship between local search length and number of local optima, which is well-studied in the single-objective case [5] and in the case of PLO-solutions [14], also applies to PLS and PLO-sets. Our results clearly show that shorter PLS lengths typically correspond to lower quality results (and hence, more difficult instances). A precise estimation of this relationship in the case of PLO-sets would require to determine the exact number of PLO-sets for a given instance.

From an algorithm design point of view, this work helps to better capture the relation between running time and approximation quality according to the problem instance characteristics. From a theoretical point of view, it would be interesting to understand precisely the relationships between the PLO-sets obtained by each archiving method. Moreover, it is clear to us that there is a direct relationship between the PLO-solutions of a problem, and the number and size of PLO-sets, however, a precise formulation remains to be described. Finally, we left for future work discussing the implications of the results reported here with respect to theoretical bounds reported in the literature [2]. Finally, complementary studies on other MCOPs and larger problem instances would allow us to better understand the structure of PLO-sets for different archiving techniques.

Acknowledgments. Manuel López-Ibáñez acknowledges support from the Belgian F.R.S.-FNRS, of which he is a postdoctoral researcher.

References

1. Aguirre, H.E., Tanaka, K.: Working principles, behavior, and performance of MOEAs on MNK-landscapes. *Eur. J. Oper. Res.* 181(3), 1670–1690 (2007)
2. Bringmann, K., Friedrich, T.: Convergence of hypervolume-based archiving algorithms I: Effectiveness. In: Krasnogor, N., et al. (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pp. 745–752. ACM Press, New York (2011)
3. Drugan, M.M., Thierens, D.: Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies. *J. Heuristics* 18(5), 727–766 (2012)
4. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Comput. Oper. Res.* 38(8), 1219–1236 (2011)
5. Kauffman, S.A.: *The Origins of Order*. Oxford University Press (1993)
6. Knowles, J.D.: *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. Ph.D. thesis, University of Reading, UK (2002)
7. Knowles, J.D., Corne, D.: Bounded Pareto archiving: Theory and practice. In: Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (eds.) *Metaheuristics for Multiobjective Optimisation*. LNEMS, vol. 535, pp. 39–64. Springer, Heidelberg (2004)
8. Laumanns, M., Zenklusen, R.: Stochastic convergence of random search methods to fixed size Pareto front approximations. *Eur. J. Oper. Res.* 213(2), 414–421 (2011)
9. López-Ibáñez, M., Knowles, J., Laumanns, M.: On sequential online archiving of objective vectors. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) *EMO 2011*. LNCS, vol. 6576, pp. 46–60. Springer, Heidelberg (2011)
10. Lust, T., Teghem, J.: Two-phase Pareto local search for the biobjective traveling salesman problem. *J. Heuristics* 16(3), 475–510 (2010)
11. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (eds.) *Metaheuristics for Multiobjective Optimisation*. LNEMS, vol. 535, pp. 177–200. Springer, Heidelberg (2004)
12. Paquete, L., Schiavinotto, T., Stützle, T.: On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research* 156, 83–97 (2007)
13. Paquete, L., Stützle, T.: A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *Eur. J. Oper. Res.* 169(3), 943–959 (2006)
14. Verel, S., Liefooghe, A., Jourdan, L., Dhaenens, C.: On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives. *Eur. J. Oper. Res.* 227(2), 331–342 (2013)
15. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evol. Comput.* 7(2), 117–132 (2003)

Racing Multi-objective Selection Probabilities

Gaetan Marceau-Caron^{1,2} and Marc Schoenauer²

¹ Thales Air Systems, Rungis, France

² TAO Project, INRIA Saclay and LRI Paris-Sud University, Orsay, France
{marc.schoenauer,gaetan.marceau-caron}@inria.fr

Abstract. In the context of Noisy Multi-Objective Optimization, dealing with uncertainties requires the decision maker to define some preferences about how to handle them, through some statistics (e.g., mean, median) to be used to evaluate the qualities of the solutions, and define the corresponding Pareto set. Approximating these statistics requires repeated samplings of the population, drastically increasing the overall computational cost. To tackle this issue, this paper proposes to directly estimate the probability of each individual to be selected, using some Hoeffding races to dynamically assign the estimation budget during the selection step. The proposed racing approach is validated against static budget approaches with NSGA-II on noisy versions of the ZDT benchmark functions.

Keywords: Multi-Objective Evolutionary Optimization, Hoeffding Races, Uncertainly Handling, Noisy Multiobjective Optimization.

1 Introduction

Uncertainty handling is an important aspect of optimization since it concerns most, if not all, real-world applications. Optimizing uncertain objectives aims at taking into account modeling inaccuracies, measurement errors from sensors, or prediction errors, that will interfere with the beliefs of a decision maker about the environment. Therefore, optimization under uncertainty must include some mechanisms that ensure, one way or another, that the proposed solutions are effective, according to the user's point of view w.r.t. optimality. And whereas several definitions of such effectiveness can occur in the simplest case of a single objective, the complexity of optimizing multiple uncertain objectives increases drastically with the number of objectives.

The general framework of this work is that of multi-objective optimization in uncertain context. The degrees of freedom of the decision maker are the variables of the optimization problem, defined on the decision space, and observables are some responses of the system when setting these variables. However, the same setting will result in different responses every time it is used, and the output of the system thus defines a probability distribution, conditionally dependent on the decision variables. The goal of the optimization process is then to find the values of the decision variables that will optimize some statistics on this probability distribution. The choice of these statistics depends on the user's goal and preferences.

The average or the median are common choices, though probably sometimes only because of the lack of efficient methods to handle other statistics. For instance, risk-averse users will prefer to minimize the consequences of the worst outcomes, while risk-affine users will maximize their possible “profit” even if it comes at high risk, optimizing the *value at risk* for a given risk level.

Except when the type of noise is known –a totally unrealistic hypothesis– a common way to compute the desired statistics is to sample the fitness of each individual as many times as necessary to obtain a good estimation thereof, and the amount of computation per individual is user-defined, uniformly over the individuals and the generations. In the single-objective framework, an alternative has been proposed, using the idea of *races* [6], minimizing the number of re-evaluations while keeping a high confidence level on the results. But only limited attempts have been made in the multi-objective framework (see Section 2).

The approach proposed in this paper, *Racing Selection Probability* (RSP), is an attempt, in the multi-objective case, to dynamically decrease the number of sampling of all individuals by applying the principles of Hoeffding races directly on the estimation of the probability of being selected for an individual, using bounds on the behavior of that probability to decide as early as possible when to definitely select or discard an individual, for a given confidence level. Bounds on any statistics can be used, and straightforwardly embedded in any unmodified EMOA, thus allowing to handle any preference of the user. Furthermore, any type of noise can be handled that way.

The paper is organized the following way. Section 2 briefly surveys state-of-the-art methods for uncertainty handling in Evolutionary Multi-Objective Optimization. Section 3 introduces the Hoeffding’s inequality used in RSP. Section 4 presents experimental results on perturbed ZDT test functions (with different types of noise) where RSP is compared to the two basic noise-handling methods, the implicit and static averaging. More details and results are available in the corresponding Technical Report¹.

2 Uncertainty Handling in Multi-objective Evolutionary Optimization

The context of this work is that of Multi-Objective Optimization with Uncertainty. On the space of decision variables \mathcal{X} , several conflicting objectives f_1, \dots, f_k are defined (to be minimized, w.l.o.g.), and, as discussed above, the outcome of any given setting of the decision variables is a probability distribution \mathbf{f} over the objective space $\mathcal{F} \subset \mathbb{R}^k$ that depends on the values of the variables and on some additional unknown external random variable ε , aka noise. In particular [1], there is no “true” value of the objectives to which some random noise is added. Formally [11,12], the Multi-Objective Noisy Optimization Problem (MNOP) can be written as

$$\min_{\mathbf{x} \in \mathcal{X}} (\mathbf{f}|\mathbf{x}, \varepsilon) = (f_1, \dots, f_k|\mathbf{x}, \varepsilon) \quad (1)$$

¹ <http://hal.inria.fr/hal-01002854/en>

where \mathbf{f} is a random variable taking values in \mathcal{F} , and each f_i is a real-valued random variable, a coordinate of \mathbf{f} .

Even in the single objective case, the minimization of a random variable does not make much sense. So the user must complete the problem definition by providing some preferences through some statistics over that random variable (e.g., minimizing the mean, the median, the 5% percentile, the variance with constraint on the mean, ...). The situation is the same in the multi-objective case, except that there doesn't exist any total order on the samples of the random variable of interest. In the deterministic case, Pareto dominance has proved useful, and the notion of Pareto front is accepted as a way to describe interesting solutions of the multi-objective problem at hand. In particular, several multi-objective optimization algorithms have been proposed, among which Evolutionary Multi-Objective Algorithms (EMOAs) (see e.g., [13]). And because uncertainty is ubiquitous in real-world problems, MNOPs have also been well studied, though not always with such a degree of generality.

2.1 Previous Work

A first approach is to port to multi-objective context the single-objective *static averaging* techniques, that re-evaluate every individual N times at each generation (also called *implicit averaging* if $N = 1$).

Several works consider the specific case of additive noise of known type: the random variable ($\mathbf{f}|\mathbf{x}, \epsilon$) is of the form $\mathbf{g}(x) + \epsilon$ for some function $\mathbf{g}(x)$ and some partly known noise ϵ . Depending on the form of ϵ , approximation of the probabilistic dominance (probability that an individual Pareto-dominates another one) can sometimes be computed at low computational cost. In [10], the noise is supposed bounded, and exact calculations are done for uniform noise; In [7], the noise is assumed Gaussian with known variance (that can be computed off-line from static samples). This work is extended in [5] to the case of unknown (and non-uniform) variance. Later, [4] proposed another way to compute the probability with more general hypotheses, but going back to using a fixed number of samples (15 in experiments). In any case, it is clear that the hypothesis of a known type of noise is highly unrealistic in practice.

An approach that is specific to indicator-based algorithms is proposed in [1], that does not make any hypothesis about the noise and uses the general model of Equation 1: the indicator ϵ^+ is approximated using averages (over 5 samples), and is used within the environmental selection. However, the problem being solved there is the minimization of the expectation of the indicator at hand (w.r.t. some reference set), that cannot be adapted to the user's preferences.

Several works propose different approaches to probabilistic dominance for the general MNOP (equation 1). Pareto Dominance in Uncertain environments (PDU) [11] uses the convex hull of a fixed number of samples (10 in the paper) to estimate both the mean and its uncertainty. In [12], PDU evaluates the certainty of the mean using quartiles on each dimension, and some races are run for each objective, from [6], with confidence 0.0001 and maximum race length 15. This latter work however assumes that the noise distribution is symmetrical, and Suzuki and collaborators

propose another Pareto Dominance operator that does not need that hypothesis [2] using a CPU-expensive SVM construct over the samples; [8] improves the method using a non-parametric Mann-Whitney U-test. However, both works use a fixed number of samples (resp. 30 and 20) to estimate the dominance operator.

In [9], six different resampling approaches are compared. All but one use some absolute criteria that only depend on some statistics on the previous samples and the individual at hand to decide on early stop of the resampling procedure and derive an estimation of the mean of the sample with known confidence. That mean is then used as the fitness in a standard EMOA. The last procedure (termed OCBA) is the closest to RSP proposed here, in that it makes the minimal global sampling allocation to estimate the confidence in a partition of the population into a non-dominated and a dominated sets. However, the calculation of the confidence assumes Gaussian noise on all objectives.

2.2 Discussion, and Rationale for RSP

Our goal is to design, within a given EMOA, an approach that will limit the number of resampling while preserving some confidence on the resulting Pareto-based selection, for a wide range of statistics describing the user's preferences, and without any requirement on the type of noise. Most of the works listed above, however, use a fixed user-define resampling budget (except [9] and [12]). Furthermore, either they derive estimations of the mean of a sample with some confidence interval – and this does not allow to derive confidence bounds on the comparison between those means (except in specific cases, e.g. Gaussian distributions); or they do derive probabilistic Pareto dominance, with known confidence, but omit the second component of Pareto-based selection, the diversity preserving mechanism (the case of indicator estimation [1] is different, but strictly limited to indicator-based EMOAs).

The idea of RSP borrows from [6], like [12] cited above, is using Hoeffding races² to decrease the number of resampling while nevertheless guaranteeing some level of confidence on the statistic at hand. But contrary to the works above (including [12]), *Racing Selection Probability*, as its name claims, will perform the race on the probability of an individual to be selected by the selection mechanism of the chosen EMOA.

3 Racing Selection Probability

Let us assume some selection procedure in an existing MOEA (e.g., non-dominated sorting + crowding distance for NSGA-II [3]) that aims at selecting μ individuals out of a population of size λ . The basic idea of RSP, inspired by [6], is to estimate, for any individual i , with as few samples of fitnesses as possible thanks to Hoeffding bounds, the probability p_i^{sel} that i will be selected.

² [6] also advocates Bernstein races when the range of values is not know – which is not the case here. Hence Bernstein races will not be mentioned here.

Hoeffding’s inequality states that, for any random variable X with range width R , and for any confidence level $1 - \delta$, the absolute difference between the expectation and the empirical mean computed using t samples is upper-bounded by $R\sqrt{\log(2/\delta)/2t}$.

Every time all λ individuals are resampled, the standard selection procedure of the EMOA at hand is applied to the current sample, determining the selected μ ones. This results in a new sample for probabilities p_i^{sel} . For any δ , lower and upper values for all p_i^{sel} can be computed at confidence level $1 - \delta$, thanks to Hoeffding’s bound applied to p_i^{sel} . Any individual i whose lower bound for p_i^{sel} is larger than the upper bound of at least $\lambda - \mu$ other p_k^{sel} is definitely selected and leaves the race. Symmetrically, any individual i whose upper bound for p_i^{sel} is smaller than the lower bound of at least μ other p_k^{sel} is definitely discarded and leaves the race. Remain in the race the uncertain individuals, and only those are resampled again at next iteration. The race ends when either μ individuals are definitely selected, or $\lambda - \mu$ individuals are definitely discarded, or some maximum number of resampling T_{Max} have been done. In the latter case, selection is made without Hoeffding guarantee. Nevertheless, the proposed procedure allows to quickly select the most promising individuals with given confidence. Note that each selection step can also be done on some statistic for each individual given the past t samples. This will be illustrated in Section 4 where variants using the average or the median will be used, instead of the most recent sample. These variants will be termed RSP_{AVG} and RSP_{MED} respectively, the variant that does not use any statistic being denoted by RSP_I .

There are however some specificities to the multi-objective context. First, the selection step usually involves the whole population, be it indicator based, or using some diversity secondary criterion. Hence the individuals that have left the race should nevertheless be taken into account for the next selection steps - but without being themselves re-evaluated, of course. A bootstrap procedure is used here, to mimic an ever growing sample without any resampling. Note that bootstrap is also used to avoid resampling individuals that have not been modified by the variation operators from one generation to the next.

Finally, it might be beneficial to detect early that some race will not end before the maximum number of samples because of actual ties between individuals that remain in the uncertain set. Here, when the sum of absolute pairwise differences of the empirical mean of the p_i^{sel} becomes lower than a given threshold called *Proximity Threshold*, the race stops and the μ best individuals according to the current selection policy are returned.

4 Experimental Results

4.1 Experimental Conditions

Five methods have been experimentally compared: the implicit averaging, and two variants of the static sampling, whether the average or the median of the samples is used for the selection (see Section 2.1); and 3 variants of RSP, whether the last sample, the average or the median of the previous samples are used in the

Table 1. Parameters for (top to bottom) benchmarks and noise; static sampling; RSP

ZDT Functions:	{1,2,3,4,6}	Number of runs:	25
Deterministic(DE):	Dirac Delta Function	Gaussian noise(GA):	$0.25 * \mathcal{N}(0, \mathbb{I})$
Cauchy Noise(CA):	(0, 0.25)	Gumbel noise(GU):	(2, $2 \ln(\ln(2))$)
Population Size:	100	Nb Eval.:	{100k,500k}
SBX Crossover:	$p_c = 1.0, \eta = 20$	Polynomial Mut.:	$p_m = 1/ \mathcal{X} , \eta = 20$
Confidence Level:	{0.25, 0.95}	Proximity Thres.:	0.5
Sampling Budget:	{5, 10, 15, 20, 30, 50}	Estimators:	{None, AVeraGe, MEDian}

selection (see Section 3). All RSP variants have been implemented within NSGA-II with standard SBX crossover and polynomial mutation. A common parameter of static sampling and RSP is the *Sampling Budget*, that will denote the fixed number of samples for each individual in the static case, and the maximum length of the races in RSP. RSP also requires a *Confidence Level* and the *Proximity Threshold* (see previous Section 3).

The testbench is based on the classical ZDT suite, used either as is (deterministic setting), or with known additional noise: Gaussian noise, that should favor the average estimator compared to the median estimator, the empirical average being the minimum-variance unbiased estimator of the expectation of a normal distribution with unknown mean and variance; Cauchy noise, that has an infinite mean, hence the mean estimator should be perturbed because of the outliers; and Gumbel noise, an asymmetrical distribution with finite moments that is used in extreme value theory to simulate rare events (its location parameter is chosen in order to center the median).

The goal of the experiments is to study the impact of the two parameters *Sampling Budget* and *Confidence Level*, and possibly their interaction, e.g., if the required confidence level is too high, all races will reach the maximum budget, and RSP amounts to static sampling. All parameter values (for the algorithms and the noise models) that have been used for these experiments are listed in Table 1). All runs were limited to 100k evaluations, except ZDT6 (500k), and 25 independent runs were run for each parameter setting. Due to space limitation, results on ZDT2 are not shown, but are quite similar to those of ZDT4.

4.2 Results

The performances are compared using the *difference hypervolume* indicator w.r.t. the real Pareto front, on the normalized objective space. The normalization is done with respect to the Nadir point, computed from the union of the exact Pareto front and every point generated by each algorithm for a given function and a given noise. Statistical significance is attested by p-values of the Wilcoxon signed-rank test. Pisa performance assessment tools (<http://www.tik.ee.ethz.ch/sop/pisa>) was used to compute the hypervolumes.

Each plot of the following figures summarizes the results obtained by all algorithms on one function with one type of noise (or no noise at all): each plot displays several boxplots, each boxplot represents the statistics of the 25 hypervolume

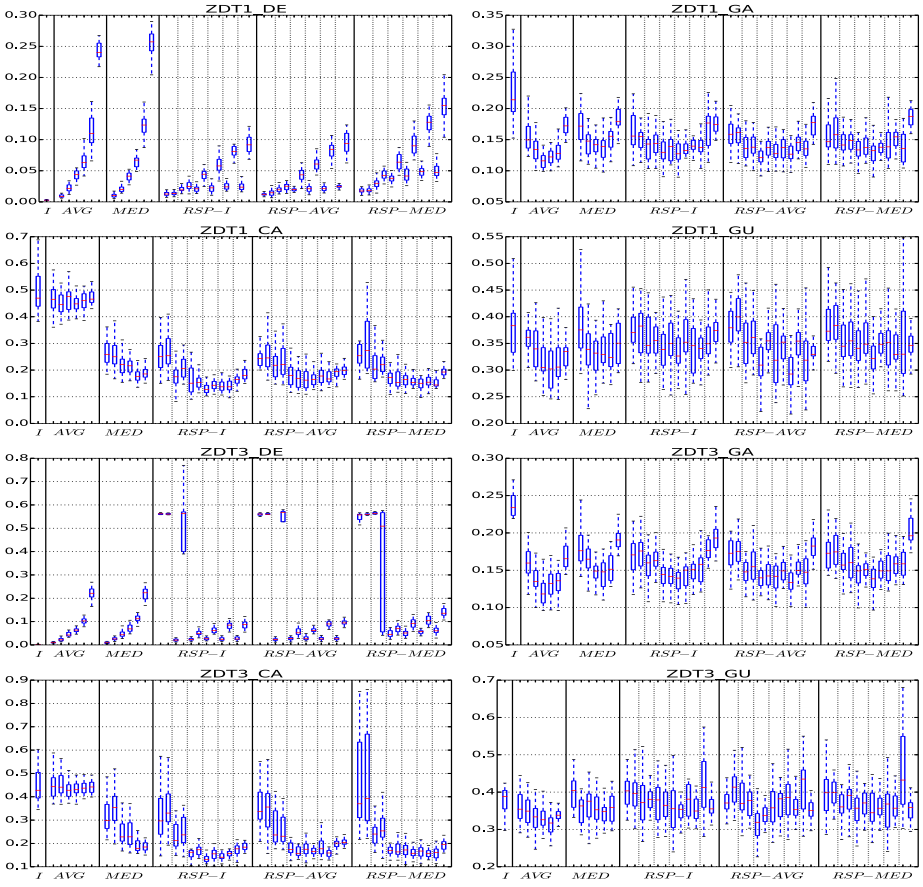


Fig. 1. Results for ZDT1 (4 top plots) and ZDT3 (4 bottom plots). See Section 4.2.

values at the end of each of the 25 runs for the corresponding setting. Each plot is divided into six regions. First boxplot is that of the implicit averaging I. Next 2 regions give the results of the static sampling (resp. *AVG* and *MED*), and display 6 boxplots each, corresponding to the 6 *Sampling Budget* values of Table 1. Next 3 regions give the results for RSP_I , RSP_{AVG} and RSP_{MED} resp. For each region, there are 6 subregions (the 6 values of *Sampling Budget*) with two boxplots each, one for each confidence level (25%, 95%).

4.3 Discussion

First of all, in the **deterministic case**, the results of implicit averaging assesses that the total budget of 100k samples is sufficient for NSGA-II to find a good approximation of the Pareto front. Furthermore, as expected, the performance of static resampling using an estimator degrades with the *Sampling Budget*, as more and more samples are wasted on the (sometimes useless) estimation of the statistic. In the same situation, RSP is able to detect the low (!) uncertainty

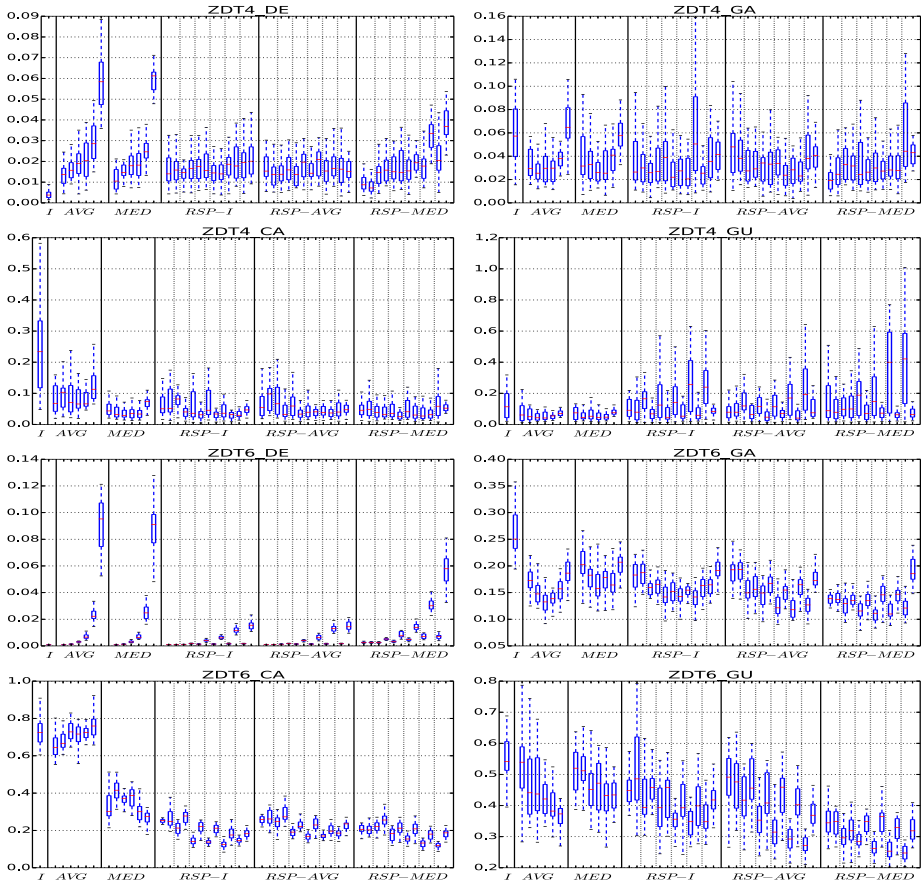


Fig. 2. Results for ZDT4 (4 top plots) and ZDT6 (4 bottom plots). See Section 4.2.

and to stop the race early, at least when using a *Confidence Level* of 25%. A *Confidence Level* of 95% can sometimes, on the other hand, lead to a similar degradation than in the static setting. The anomalies in that respect for RSP on ZDT3 (discontinuous front) for small *Sampling Budget* (5 and 10) might come from races that stop too early with all selected individuals in the same component of the front.

On the **noisy instances**, implicit averaging does not perform very well compared to the other uncertainty handling approaches. Surprisingly, even if the medians are higher, the spread of the performances is not greater than the other approaches, excepted for ZDT4-CA. It can be due to the fact that without any uncertainty handling approach, the probability that every individual of the population is good or bad is small and so, at the population level, the performance does not vary so much from one run to another.

Beside, implicit averaging is comparable to AVG in case of Cauchy noise, for all functions but ZDT4: choosing by default the mean (a common choice) can

lead to poor results when the distribution of the noise is unknown. Using RSP seems to mitigate this effect, probably because it uses the probability of survival instead of the estimator of the mean.

Comparing, for each noisy function, the best configurations of RSP and static sampling leads to the following considerations: the results are statistically equivalent for all cases of noisy ZDT2 and ZDT4; RSP is significantly better (p-value $< 10^{-5}$) than static sampling in 5 cases (the 3 noisy ZDT6, and ZDT1 and 3 with Cauchy noise), is slightly worse (p-value in $[0.01, 0.1]$) in 2 cases (both ZDT1 and 3 with Gaussian noise), and both approaches are equivalent (p-value > 0.1) on the remaining 2 cases (both ZDT1 and 3 with Gumbel noise). On ZDT1 and 3 with Gaussian noise, static sampling with averaging performs best: this is most probably related to the fact that AVG is based on the minimum-variance unbiased estimator, while RSP_{AVG} uses it indirectly to estimate the probability of survival.

Regarding the choice of estimator, racing seems to decrease the impact of the average vs median issue. Indeed, when using static sampling, average performs slightly better than median for Gaussian and Gumbel noises, whereas median is consistently and significantly better when facing Cauchy noise. On the opposite, all 3 variants of RSP perform in general similarly over all problems. In particular, the no-estimator, RSP_I , performs as good as both others on most problems. This is good news, as it gives hope that the proposed racing approach might perform well with a lot of estimators, allowing the user to actually choose his favorite without having to care about the optimization algorithm in that respect.

5 Conclusion and Perspective

RSP is a general approach to uncertainty handling in existing EMOAs. It uses a (μ, λ) Hoeffding race at a given confidence level, inspired by [6], though applied directly on the selection probabilities of the individuals in the population. It is agnostic w.r.t. the selection method, and hence can accommodate any user preference that could be carried by the algorithm selection.

First experimental results within NSGA-II on noisy versions of ZDT benchmarks, indicate that this path is worth following for future research: RSP performs significantly better than implicit averaging or static sampling in many situations, and never performs significantly worse. It is less sensitive to the *Sampling Budget* parameter, especially for small (on zero) levels of noise, and surprisingly almost insensitive to the choice of the estimator. On the other hand, it is very sensitive to the *Confidence Level* of the races. However, these partial conclusions should be sustained by deeper analyses and validated by more experiments, with different levels of non-homogeneous noise, and other test functions from real-world problems.

The main perspectives for further work are to couple RSP with other EMOAs such as SPEA-2, IBEA and HYPE, in order to study the interaction between racing and the indicator function. Also, RSP should also be compared to more sophisticated uncertainty handling methods (see Section 2). It is also mandatory to test other estimators within RSP, as well as different noise models and different noise intensities. On the more fundamental side, it should be possible

to better understand the intricate relationship between estimating the selection probability and directly estimating the objective values.

A longer term research track is to come up with some adaptive procedure to dynamically tune the *Sampling Budget* and, maybe more importantly, the *Confidence Level*. Indeed, it is clear from the present results that this latter parameter has a strong effect on the performance of the algorithm and should be fixed carefully. In the case where its optimal value varies over the decision space, only adaptive tuning can perform well on most functions. To conclude, we feel that the use of RSP in EMOA is a promising avenue for taking into account the decision maker's preferences and increasing the reliability and robustness of the solutions in many real-world applications.

References

1. Basseur, M., Zitzler, E.: A Preliminary Study on Handling Uncertainty in Indicator-Based Multiobjective Optimization. In: Rothlauf, F., et al. (eds.) *EvoWorkshops 2006*. LNCS, vol. 3907, pp. 727–739. Springer, Heidelberg (2006)
2. Boonma, P., Suzuki, J.: A Confidence-based Dominance Operator in Evolutionary Algorithms for Noisy Multiobjective Optimization Problems. In: *Proc. ICTAI 2009*. IEEE Press (2009)
3. Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons (2001)
4. Eskandari, H., Geiger, C.D., Bird, R.: Handling uncertainty in evolutionary multi-objective optimization: SPGA. In: *CEC 2007*, pp. 4130–4137. IEEE Press (2007)
5. Fieldsend, J., Everson, R.: Multi-Objective Optimisation in the Presence of Uncertainty. In: *CEC 2005*, pp. 243–250. IEEE Press (2005)
6. Heidrich-Meisner, V., Igel, C.: Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search. In: Danyluk, A.P., et al. (eds.) *Proc. ICML 2009*. ACM Intl. Conf. Proc. Series, vol. 382, p. 51 (2009)
7. Hughes, E.J.: Evolutionary Multiobjective Ranking with Uncertainty and Noise. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) *EMO 2001*. LNCS, vol. 1993, pp. 329–343. Springer, Heidelberg (2001)
8. Phan, D.H., Suzuki, J.: A Non-parametric Statistical Dominance Operator for Noisy Multiobjective Optimization. In: Bui, L.T., Ong, Y.S., Hoai, N.X., Ishibuchi, H., Suganthan, P.N. (eds.) *SEAL 2012*. LNCS, vol. 7673, pp. 42–51. Springer, Heidelberg (2012)
9. Siegmund, F.: *Sequential Sampling in Noisy Multi-Objective Evolutionary Optimization*. Master's thesis, School of Humanities and Informatics, Skövde (2009)
10. Teich, J.: Pareto-Front Exploration with Uncertain Objectives. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) *EMO 2001*. LNCS, vol. 1993, pp. 314–328. Springer, Heidelberg (2001)
11. Trautmann, H., Mehnen, J., Naujoks, B.: Pareto-Dominance in Noisy Environments. In: Tyrrell, A. (ed.) *Proc. CEC 2009*, pp. 3119–3126. IEEE Press (2009)
12. Voß, T., Trautmann, H., Igel, C.: New Uncertainty Handling Strategies in Multi-objective Evolutionary Optimization. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *PPSN XI*. LNCS, vol. 6239, pp. 260–269. Springer, Heidelberg (2010)
13. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation* 1(1), 32–49 (2011)

Shake Them All!

Rethinking Selection and Replacement in MOEA/D

Gauvain Marquet, Bilel Derbel, Arnaud Liefooghe, and El-Ghazali Talbi

¹ Université Lille 1, LIFL, UMR CNRS 8022, France

² Inria Lille - Nord Europe, DOLPHIN project-team, France
`{firstname.lastname}@inria.fr`

Abstract. In this paper, we build upon the previous efforts to enhance the search ability of MOEA/D (a multi-objective decomposition-based algorithm), by investigating the idea of evolving the whole population simultaneously. We thereby propose new alternative selection and replacement strategies that can be combined in different ways within a generic and problem-independent framework. To assess the performance of our strategies, we conduct a comprehensive experimental study on bi-objective combinatorial optimization problems. More precisely, we consider ρ MNK-landscapes and knapsack problems as a benchmark, and experiment a wide range of parameter configurations for MOEA/D and its variants. Our analysis reveals the effectiveness of our strategies and their robustness to parameter settings. In particular, substantial improvements are obtained compared to the conventional MOEA/D.

1 Introduction

Evolutionary multi-objective optimization (EMO) algorithms [1, 2] have been proved extremely effective in computing a high-quality approximation of the Pareto set, i.e., the set of solutions providing the best compromises between the multiple objectives of an optimization problem. In particular, decomposition-based (or aggregation-based) algorithms are gaining in popularity as an increasing number of studies is being devoted to their development [3–7]. Recently, MOEA/D [4] (Multi-Objective Evolutionary Algorithm based on Decomposition) has attracted a lot of interest; which is due to its simplicity, approximation quality, and computational efficiency. In this paper, we seek for new alternative selection mechanisms for MOEA/D at the aim of enhancing its search quality; and we focus on bi-objective combinatorial problems as a case study.

Generally speaking, MOEA/D builds upon the idea of decomposing the initial multi-objective optimization problem into several single-objective *sub-problems* by means of scalarizing functions [8] configured with different weight vectors. The most original part of MOEA/D is to define, for each sub-problem, a *neighborhood* structure containing the set of the closest sub-problems. Then, MOEA/D iterates over sub-problems and performs the three following basic steps: (i) select parents among the neighbors of the current sub-problem, (ii) generate an offspring by applying problem-specific operators, and (iii) replace neighbors' solutions if the generated offspring is better. We remark that mating selection (Step (i))

is performed exclusively among neighbors. Assuming that nearby sub-problems have similar solutions, the neighborhood size is critical for an accurate exploration/exploitation balance. Moreover, the replacement mechanism (Step (iii)) can lead to a situation where several neighbors are replaced by the *same* offspring. This can imply a loss of diversity, and likely a loss in performance. These issues have actually been addressed in [9], where two extra modifications have been introduced when dealing with complicated continuous Pareto sets. The first modification uses an extra probability parameter allowing parents to be selected from the whole population. The second one uses an extra parameter to bound the number of neighbors that can be replaced by a newly generated offspring.

In this paper, we propose new selection strategies to enhance the search ability of MOEA/D for combinatorial problems. The idea behind our strategies stems from the observation that if an offspring is allowed to replace a neighboring solution in MOEA/D, this solution is then ‘lost’ and it has no chance to get selected for reproduction in subsequent iterations. To overcome this shortcoming, we investigate an alternative perspective in MOEA/D by optimizing all sub-problems *at once*. Intuitively, every solution from the population has a more fair chance to participate in the evolution process. This allows us to propose different strategies that can be plugged in the basic version of MOEA/D. Our newly proposed strategies do not distort the basic framework of MOEA/D, neither do they induce a loss in generality nor do they introduce new extra parameters; while being fully compatible with the previous modifications introduced so far. Moreover, they are proven to exhibit substantial improvements in approximation quality when compared with basic MOEA/D and its modifications. Our performance assessment is in fact obtained as the byproduct of a thorough experimental analysis on two bi-objective combinatorial optimization problems, namely, knapsack and ρ MNK-landscapes, and by considering a broad range of configurations. In the remainder, we first recall in Sec. 2 some basic definitions as well as a brief description of MOEA/D. In Sec. 3, we describe our algorithmic contribution in designing new selection and replacement strategies for MOEA/D. In Sec. 4, we present the settings of our experimental study. In Sec. 5, we state our main experimental findings. Finally, we conclude the paper in Sec. 6.

2 Background

Definitions. A *multi-objective optimization problem* can be defined by an objective function vector $f = (f_1, \dots, f_m)$ with $m \geq 2$, and a set X of feasible solutions in the *solution space*. In the combinatorial case, X is a discrete set. Let $Z = f(X) \subseteq \mathbb{R}^m$ be the set of feasible outcome vectors in the *objective space*. To each solution $x \in X$ is assigned an objective vector $z \in Z$, on the basis of the function vector $f : X \rightarrow Z$ with $z = f(x)$. In a maximization context, a solution $x \in X$ is dominated by a solution $x' \in X$ iff $\forall i \in \{1, \dots, m\}$, $f_i(x) \leq f_i(x')$ and $\exists i \in \{1, \dots, m\}$ such that $f_i(x) < f_i(x')$. A solution $x^* \in X$ is said to be *Pareto optimal* (or *non-dominated*), if there does not exist any other solution $x \in X$ such that x^* is dominated by x . The set of all Pareto optimal solutions is the *Pareto set*. Its mapping in the objective space is the *Pareto front*.

Decomposition-Based EMO. Contrary to existing Pareto-based EMO algorithms, like NSGA-II or SPEA2, which explicitly use the Pareto dominance relation in their selection mechanism, decomposition-based EMO algorithms [10] rather seek a good-performing solution in multiple regions of the Pareto front by *decomposing* the original multi-objective problem into a number of *scalarized* single-objective sub-problems, which can be solved independently as in MsOPS [7], or in a dependent way as in MOEA/D [4]. Many different scalarizing functions have been proposed in the literature [8]. Popular examples are the weighted sum (g^{ws}) and the weighted Tchebycheff (g^{te}) functions defined below:

$$g^{ws}(x, \lambda) = \sum_{i=1}^m \lambda_i \cdot f_i(x) \quad , \quad g^{te}(x, \lambda) = \max_{i \in \{1, \dots, m\}} \lambda_i \cdot |z_i^* - f_i(x)|$$

where x belongs to the solution space, $\lambda = (\lambda_1, \dots, \lambda_m)$ is a weighting coefficient vector such that $\lambda_i \geq 0$ for all i , and $z^* = (z_1^*, \dots, z_m^*)$ is a utopian point, i.e., $\forall i, \forall x, z_i^* > f_i(x)$. g^{te} (resp. g^{ws}) is to be minimized (resp. maximized).

MOEA/D in a Nutshell. Let g be a scalarizing function and let $(\lambda^1, \dots, \lambda^\mu)$ be a set of μ uniformly distributed weighting coefficient vectors, corresponding to μ sub-problems to be optimized. For each sub-problem $i \in \{1, \dots, \mu\}$, the goal is to approximate the solution x with the best scalarizing function value $g(x, \lambda^i)$. For that purpose, MOEA/D maintains a population $P = (p^1, \dots, p^\mu)$, each individual corresponding to a good-quality solution for one sub-problem. For each sub-problem $i \in \{1, \dots, \mu\}$, a set of neighbors $\mathcal{B}(i)$ is defined with the T closest weighting coefficient vectors. To evolve the population, subproblems are optimized iteratively. At a given iteration corresponding to one sub-problem i , two solutions are selected at random from $\mathcal{B}(i)$, and an offspring solution x is created by means of variation operators (mutation and crossover). A problem-specific repair or improvement heuristic is potentially applied on solution x to produce x' . Then, for every sub-problem $j \in \mathcal{B}(i)$, if x' improves over j 's current solution p^j then x' replaces it. The algorithm continues looping over sub-problems, optimizing them one after the other, until a stopping condition is satisfied. We shall also consider the two modifications introduced in [9] to enhance MOEA/D in the context of *continuous* complicated Pareto sets. The first one allows to select a parent from the whole population with a small probability parameter $(1 - \delta)$. More precisely, when dealing with a sub-problem i , its neighborhood is set to $\mathcal{B}(i)$ with probability δ , and to the whole population P with probability $(1 - \delta)$. The second one limits by a parameter nr the number of times that an offspring x' , created when dealing with a sub-problem i , can replace solutions in the neighborhood of i .

3 Rethinking Selection and Replacement in MOEA/D

As mentioned in the introduction, MOEA/D could suffer from a lack of diversity due to the locality of its selection and replacement mechanism. We argue that this can also be caused by the fact that in MOEA/D (and its modified variants), sub-problems are optimized iteratively. In fact, since parents are selected randomly from the neighborhood of the sub-problem being processed, it might happen that

a solution with the potential of producing a good offspring, gets never selected for reproduction. Additionally, because a neighbor's solution might be replaced as soon as a better offspring is found, this solution gets actually no chance to survive in the population. To increase the chance for a solution to survive in the population, we investigate the idea of evolving the whole population *simultaneously* by optimizing all subproblems in one shot and not iteratively. This idea is depicted in Algorithm 1 and discussed more thoroughly in the following.

Algorithm 1. Our proposed framework MOEAD-**xy** ($\mathbf{x}, \mathbf{y} \in \{\mathbf{s}, \mathbf{c}\}$)

Input: $\{\lambda^1, \dots, \lambda^\mu\}$: weight vectors w.r.t sub-problems; g : a scalarizing function; $\mathcal{B}(i)$: the neighbors of sub-problem $i \in \{1, \dots, \mu\}$; $P = \{p^1, \dots, p^\mu\}$: the initial population.

```

1 while STOPPING CONDITION do
2   for  $i \in \{1, \dots, \mu\}$  do
3     if  $\text{rand}(0, 1) < \delta$  then  $B_i \leftarrow \mathcal{B}(i)$ ; /* Neighborhood Setting */
4     else  $B_i \leftarrow P$  if  $\mathbf{x} = \mathbf{s}$  then /* Selfish mating selection */
5       |  $k \leftarrow i$ ;
6     else if  $\mathbf{x} = \mathbf{c}$  then /* Collective mating selection */
7       |  $k \leftarrow \text{rand}(B_i)$ ;
8      $\ell \leftarrow \text{rand}(B_i)$ ; while  $\ell = k$  do  $\ell \leftarrow \text{rand}(B_i)$ 
9     if  $\text{rand}(0, 1) < cr$  then /* Variation operators */
10      |  $o^i \leftarrow \text{crossover}(p^k, p^\ell)$ ;  $o^i \leftarrow \text{mutation}(o^i)$ ;
11      | else  $o^i \leftarrow \text{mutation}(p^k)$  if  $o^i$  is infeasible then  $\text{repair}(o^i)$ 
12   for  $i \in \{1, \dots, \mu\}$  do  $c_i \leftarrow 0$ 
13   for  $i \in \{1, \dots, \mu\}$  do /* Environmental replacement */
14     if  $\mathbf{y} = \mathbf{s}$  then /* Selfish replacement */
15       |  $p' \leftarrow o^i$ ;
16       | if  $g(p', \lambda^i)$  better than  $g(p^i, \lambda^i)$  then  $p^i \leftarrow p'$ 
17     else if  $\mathbf{y} = \mathbf{c}$  then /* Collective replacement */
18       | shuffle( $B_i$ );
19       | for  $j \in B_i$  do
20         |  $p' \leftarrow o^j$ ;
21         | if  $c_j < nr$  then
22           | | if  $g(p', \lambda^i)$  better than  $g(p^i, \lambda^i)$  then  $p^i \leftarrow p'$ ;  $c_j \leftarrow c_j + 1$ 

```

Algorithm 1 is mainly divided in two stages (lines 2 to 11 and lines 12 to 22). Contrary to MOEA/D where a single offspring is generated at each iteration, our framework is basically a $(\mu + \mu)$ -EA where the first stage consists in generating μ offsprings and the second stage consists in updating the whole population for the next round. The first stage corresponds to mating selection where *one* new offspring is created for *every* subproblem. Specifically, we consider two alternatives: (i) either the solution of the current subproblem is *always* selected to be a parent and hence included for variation ($\mathbf{x} = \mathbf{s}$), or (ii) parents are picked randomly from neighbors in the usual way MOEA/D proceeds ($\mathbf{x} = \mathbf{c}$). Moreover, every offspring is tagged with the identifier of the subproblem where it has been created. Thus, we can identify the subproblem that originated the creation of a given offspring. Only when all subproblems are treated and all μ new offspring solutions are created, the second stage of replacement occurs. In this stage, the subproblems are processed iteratively and we again consider two alternatives: (i) either the solution of a subproblem is compared to the offspring created at this subproblem ($\mathbf{y} = \mathbf{s}$), or (ii) the solution of the current subproblem is compared to the offsprings created in neighboring subproblems ($\mathbf{y} = \mathbf{c}$).

In both cases, the solution of the current subproblem gets replaced if the considered offspring shows an improvement.

Algorithm 1 is fully compatible with the baseline ideas of MOEA/D; in particular, with the variants in [9], i.e., parameters δ and nr . Due to lack of space, we omit describing all the standard aspects that are shared with MOEA/D, e.g., weights initialization, neighborhoods, update of the reference point, archiving.

To summarize, Algorithm 1 differs from MOEA/D by essentially the fact that μ offsprings for all subproblems are created at each iteration. Moreover, since two alternatives are designed for mating selection and replacement, four different variants are possible: MOEAD-**xy** with $\mathbf{x}, \mathbf{y} \in \{\mathbf{s}, \mathbf{c}\}$ — **s** (resp. **c**) refers to a Selfish (resp. Collective) strategy where a subproblem privileges its own solution (resp. its neighbors' solutions). It is worth to notice that some parameter combinations may not have any impact on some algorithm variants, e.g., nr does not have an impact on MOEAD-**ss** and MOEAD-**cs**, neither δ on MOEAD-**ss** when $cr = 0$.

4 Experimental Setup

We analyze our approach on bi-objective ρ MNK-landscapes and knapsack problems, with a broad range of instances with different structures and sizes.

ρ MNK-Landscapes. The family of ρ MNK-landscapes constitutes a problem-independent model used for constructing multi-objective multi-modal landscapes with objective correlation [11]. A bi-objective ρ MNK-landscape aims at maximizing an objective function vector $f : \{0, 1\}^n \rightarrow [0, 1]^2$. Solutions are binary strings of size n . The parameter k defines the number of variables that influence a particular position from the bit-string (the epistatic interactions). By increasing the number of variable interactions k from 0 to $(n - 1)$, landscapes can be gradually tuned from smooth to rugged. The objective correlation parameter ρ defines the degree of conflict between the objectives. The positive (resp. negative) data correlation allows to decrease (resp. increases) the degree of conflict between the objective function values. This has an impact on the cardinality of the Pareto front [11]. We investigate six random ρ MNK-landscapes for each parameter combination given in Table 1.

Knapsack. The knapsack problem is one of the most studied NP-hard problem. Given a collection of n items and a set of 2 knapsacks, the 0 – 1 bi-objective bi-dimensional knapsack problem seeks a subset of items subject to capacity constraint based on a *weight function* vector $w : \{0, 1\}^n \rightarrow \mathbb{N}^2$, while maximizing a *profit function* vector $p : \{0, 1\}^n \rightarrow \mathbb{N}^2$. More formally, it can be stated as:

$$\max \sum_{j=1}^n p_{ij} \cdot x_j \quad ; \quad \text{s.t.} \quad \sum_{j=1}^n w_{ij} \cdot x_j \leq c_i \quad i \in \{1, 2\} \\ x_j \in \{0, 1\} \quad j \in \{1, \dots, n\}$$

where $p_{ij} \in \mathbb{N}$ is the profit of item j on knapsack i , $w_{ij} \in \mathbb{N}$ is the weight of item j on knapsack i , and $c_i \in \mathbb{N}$ is the capacity of knapsack i . We consider the standard instances proposed in [12], with random uncorrelated profit and weight integer values from $[10, 100]$, and where capacity is set to half of the total weight of a

Table 1. Parameter setting

		ρ MNK-landscapes m = 2, n = 128	Knapsack m = 2		
		$\rho \in \{-0.7, 0.0, 0.7\}, K \in \{4, 8\}$	n = 250	n = 500	n = 750
pop size	μ	64, 128, 256	150	200	250
neighborhood size	T	4, 8, 16, 32	10, 20, 30		
max. number of replacements	nr	1, 2, 3, 4, ∞	2, 4, 8, 10, ∞		
neighborhood probability	δ	0.9, 1.0			
crossover rate	cr	0.0, 0.9, 1.0			
scalarizing function	g	weighted sum (g^{ws}), weighted Tchebycheff (g^{te})			
stopping condition		10^6 evaluation function calls	10^6 repair procedure calls		

knapsack. Thirty different random problem instances are investigated for each parameter combination given in Table 1. Moreover, we use the same advanced *weighted* repairing procedure to handle constraints as in MOEA/D [4].

Parameter Setting. Table 1 shows the parameter settings investigated in our study. We consider the effect of the population size (μ), the neighborhood size (T), the maximum number of neighboring solutions replaced (nr), the probability to select a parent outside of a neighborhood ($1 - \delta$), the scalarizing function (g), and the crossover probability (cr). The stopping condition is set to 10^6 evaluation (resp. repair) calls for ρ MNK-landscapes (resp. knapsack). Standard MOEA/D [4, 9] is considered, together with our four variants. We use a bit-flip mutation (where each bit is independently flipped with a rate $1/n$) and one-point crossover. The crossover probability parameter (cr) allows us to appreciate the impact of the variation operator, from a pure randomized local search algorithm ($cr = 0.0$) to a conventional genetic algorithm ($cr = 1.0$). The initial population is generated randomly. An unbounded archive of all non-dominated solutions is maintained with all the approaches. All algorithms have been executed under comparable conditions and share the same base components. Overall, we tested 15918 different configurations, each one executed 30 times. Due to space limitations, we only highlight a subset of settings allowing us to state our findings.

5 Experimental Analysis

Algorithm Comparison. We follow the performance assessment protocol proposed by [13] using the hypervolume difference and multiplicative epsilon indicators [14]. The hypervolume difference indicator (I_H^-) gives the difference between the portion of the objective space that is dominated by the Pareto set approximation and some reference set. The reference point is set to the worst value obtained over all approximations, and the reference set is the best-found approximation over all tested configurations. The epsilon indicator (I_ϵ^\times) gives the minimum multiplicative factor by which the approximation found by an algorithm has to be translated in the objective space to weakly dominate the reference set.

Due to space limitations, we shall not focus on eliciting the best configurations; but give an overview of the differences between algorithms and their robustness to parameters. A non-exhaustive set of results is shown in Tables 2 and 3. First, notice the strong impact of the scalarizing function (g^{ws} or g^{te}) on

performance and its dependency on the considered problem. Overall, MOEAD-**sc** and MOEAD-**cc** are highly competitive and exhibit the most appealing behaviors. For knapsack, these two variants perform similarly to MOEA/D. This can be explained by the relative strength of the repair function, and also by the shape of the Pareto front for knapsack problems, which is relatively easy to approximate. For ρ MNK-landscapes with different structures, substantial improvements are reported, independently of the parameter setting. Actually, MOEAD-**ss** is also found to be competitive, but only when the crossover is activated. This is because MOEAD-**ss** degenerates to a multiple independent search in this case; and thus it is more likely trapped into independent local optima. At the opposite, MOEAD-**sc** and MOEAD-**cc** are able to adequately use information from neighbors, even when only a mutation operator is considered.

The previous discussion is in general valid when conducting an “anytime” analysis as is illustrated in Fig. 1 rendering the convergence of competing algorithms. We see that all algorithms are able to make improvements, with MOEAD-**sc** and MOEAD-**cc** being consistently better than MOEA/D. These results confirm that shaking many solutions at once can serve the approximation quality till the early stages of the search process. We also remark that MOEAD-**sc** and MOEAD-**cc** are more systematically improving upon MOEAD-**ss** for test instances having conflicting objectives; whereas MOEAD-**ss** is able to outperform its competitors as the objective correlation gets higher. Notice in fact that our strategies induce different intensification/diversification trade-offs both at the local level of every single-objective scalarized subproblem; but also at a more global level when considering the whole approximation set. When a selfish (resp. collective) mating selection is considered, the probability that a solution in the population gets selected for reproduction is 1 (resp. $1 - (1 - 1/T)^T$). Roughly speaking, this means that all our strategies imply diversified offsprings since no solution in the current population gets replaced before exploring its potential. At the replacement stage, if a collective strategy is adopted, then the single-objective search at every subproblem is intensified since the probability that a locally improving solution can be found is higher. But this might increase the number of copies in the current approximation set. When a selfish replacement is considered, it is more likely that the number of copies is minimized; but at the price of delaying the advance of the population towards the front. For correlated objectives, and since the front is not too large, it is sufficient that only few solutions are able to approach the front in order to get good overall performance. Thus, a selfish replacement can be accurate. This is not the case for anti-correlated objectives where both the local improvements at every subproblem and the global spread of solutions is crucial. This explains the relative performance of our strategies depending on the characteristic of the tackled problem.

Impact of Parameters. From Tables 2 and 3, we can already extract some interesting observations on the impact of parameters, e.g., notice the differences between g^{ws} and g^{te} . Further observations from our data are sketched in Fig. 2, where only MOEA/D, MOEAD-**sc** and MOEAD-**cc** are highlighted. First (Fig. 2 left), we confirm the positive impact of small values of parameter nr [9] on

Table 2. Representative subset of configurations w.r.t I_H^- and I_ε^x and ρ MNK-landscapes ($\mu = 128, T = 8, \delta = 1.0$) at termination. For each row, the numbers indicates how many algorithms (over the other 15 configurations given in columns) outperforms the configuration under consideration with a statistical confidence level of 0.05 (the lower, the better).

		g^{Te}										g^{ws}												
		MOEA/D		MOEAD-SS		MOEAD-SC		MOEAD-CS		MOEAD-CC		MOEA/D		MOEAD-SS		MOEAD-SC		MOEAD-CS		MOEAD-CC				
		∞	2	-	∞	2	-	∞	2	-	∞	2	∞	2	-	∞	2	-	∞	2	-	∞	2	
1.0	cr	ρ	I_H^-																					
		4	2	2	8	0	0	5	0	0	4	5	0	6	2	7	5	6						
		8	3	1	7	0	0	6	1	0	11	7	6	9	6	9	9	8						
		4	2	1	0	0	0	2	0	0	2	0	0	0	0	0	0	2						
		8	6	0	0	0	1	6	0	0	8	7	3	8	9	12	6	0						
		4	14	10	0	2	2	2	2	1	1	2	0	2	2	2	2	1						
		8	2	2	0	2	1	2	2	2	2	1	0	5	2	1	2	1						
			I_ε^x																					
		4	8	8	1	0	0	1	0	0	0	5	0	5	0	8	2	3						
		8	5	5	0	0	0	5	5	0	7	6	0	9	5	7	7	5						
		4	2	1	0	0	0	1	0	0	0	0	1	1	1	1	0	1						
		8	4	0	0	0	1	3	0	0	9	9	6	9	9	10	7	7						
		4	15	4	0	2	2	2	2	0	2	2	0	2	2	2	3	0						
		8	1	1	0	2	0	1	1	2	2	1	1	6	2	1	4	2						
0.0			I_H^-																					
		4	0	4	15	0	0	4	0	0	6	5	13	5	5	12	6	6						
		8	1	1	15	1	0	6	0	0	9	6	6	6	6	10	6	8						
		4	3	0	15	0	0	0	0	0	3	0	11	0	0	0	0	0						
		8	0	0	15	0	0	4	0	0	6	8	6	8	5	6	6	5						
		4	9	1	9	1	0	1	1	0	3	1	2	0	0	0	0	0						
		8	0	1	0	4	0	1	1	1	2	1	0	1	1	1	4	4						
			I_ε^x																					
		4	6	8	10	1	1	1	1	1	6	2	0	4	5	10	2	4						
		8	3	5	5	1	1	5	1	1	8	5	0	7	5	10	5	9						
		4	3	1	13	0	0	1	0	0	1	1	1	0	0	1	1	1						
		8	0	0	8	0	0	0	0	0	8	8	6	8	8	8	8	8						
		4	6	0	4	0	0	1	0	0	3	2	3	0	0	0	0	0						
		8	0	1	0	3	0	1	0	0	2	0	0	2	2	0	10	6						

Table 3. Relative performance of a representative subset of configurations for knapsack ($T = 20, \delta = 0.9$). Metrics similar to those in Table 2 are reported.

		g^{Te}										g^{ws}												
		MOEA/D		MOEAD-SS		MOEAD-SC		MOEAD-CS		MOEAD-CC		MOEA/D		MOEAD-SS		MOEAD-SC		MOEAD-CS		MOEAD-CC				
		∞	2	-	∞	2	-	∞	2	-	∞	2	∞	2	-	∞	2	-	∞	2	-	∞	2	
0.9	cr	N	I_H^-																					
		250	8	8	7	8	8	8	8	8	0	0	0	0	0	0	0	0						
		500	8	8	8	8	8	13	8	8	0	0	0	0	0	0	0	0						
		750	8	8	8	8	8	14	8	8	0	0	0	0	0	0	0	0						
			I_ε^x																					
		250	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0						
		500	8	8	7	7	7	11	7	7	0	5	0	0	4	1	4	5						
		750	4	3	6	4	3	7	3	2	5	0	0	5	7	7	0	8						

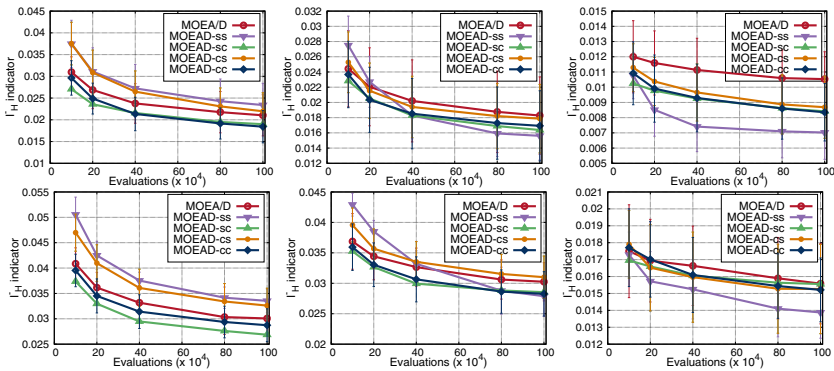


Fig. 1. Convergence plots w.r.t. hypervolume difference and ρ MNK-landscapes. Columns are respectively for $\rho \in \{-0.7, 0.0, 0.7\}$. Rows are respectively for $K \in \{4, 8\}$. Results are for $\mu = 128$, $T = 8$, $\delta = 1.0$, $cr = 1.0$, $nr = 0$ and g^{te} .

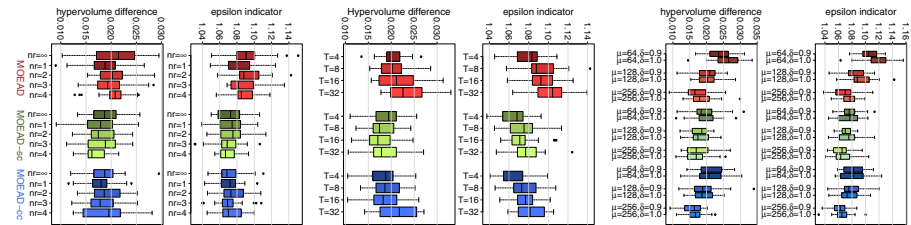


Fig. 2. Impact of parameters. The two subfigures in the left (resp. middle, right) show the impact of nr (resp. T , μ combined with δ) as shown in the vertical axis for every algorithm. The boxplots are w.r.t. the indicator depicted at the horizontal axis. Results are for a ρ MNK-landscape with $\rho = -0.7$ and $K = 4$. Whenever not explicit: $\mu = 128$, $T = 8$, $cr = 1.0$, $\delta = 1.0$, $nr = 2$ and g^{te} .

the performance of MOEA/D for combinatorial problems. However, it was not always clear what is the best value to choose independently of the other parameters, e.g., the recommended value of 2 is in fact accurate, but not always optimal. Also, the impact of parameter nr on our strategies is rather mitigated. Although we found that it could often bring improvements, the impact was not pronounced compared to the case of MOEA/D. For neighborhood size T (Fig. 2 middle), we found that, contrary to knapsack, small values of T are interestingly more accurate for ρ MNK-landscapes. We attribute this to the influence of the crossover operator which, combined with the repair mechanism, does enable to find high-quality solutions for knapsack. However, for ρ MNK-landscapes, it is more likely that the crossover diversifies the search too much when considering parents in a relatively large neighborhood. Finally (Fig. 2 right), parameter δ used for neighborhood selection is confirmed to have a positive impact on the performance. However, we find that another parameter has even more effect; namely the population size μ and especially for anti-correlated instances. We attribute this to the fact that, when the Pareto front gets larger, it is beneficial to increase the population size in order to distribute the population efficiently.

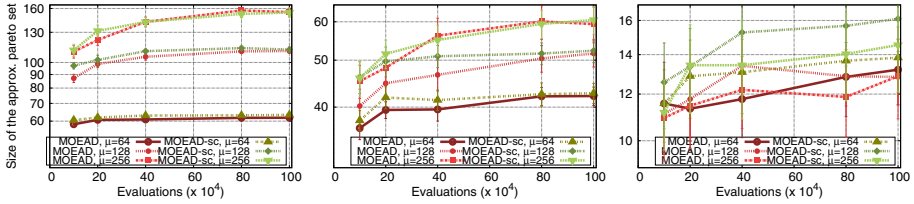


Fig. 3. Solution diversity. The y-axis gives the number of different solutions in the population (in log-scale). Results are for a ρ MNK-landscape with $\rho \in \{-0.7, 0.0, 0.7\}$ and $K = 8$ ($\mu = 128$, $T = 8$, $cr = 1.0$, $\delta = 1.0$, $nr = 2$ and g^{te}).

Diversity Issues. We conclude our analysis by illustrating in Fig. 3 the size of the Pareto set approximation extracted at different iterations from the population (without the archive) for MOEA/D and MOEA/D-sc. In fact, we were able to observe that our strategy tends to maintain more spread solutions and to distribute them efficiently over the weight vectors, independently of the population size μ . We argue that this is a key feature of why our variants are able to exhibit better performance over MOEA/D. Of course, this is not the only ingredient for optimal anytime performance; but it contributes much in finding a high-quality approximation, especially for conflicting objectives.

6 Conclusions and Perspectives

We introduced a framework incorporating four strategies to deal with selection and replacement in MOEA/D. Our experimental results show that substantial improvements can be obtained. Moreover, our study opens new possibilities for improving the design of decomposition-based algorithms in several perspectives. Firstly, one can wonder whether more general $(\mu + \lambda)$ -EA can be embedded in our framework. Secondly, we think that our framework opens the road to high-quality local search-based MOEA/D variants for combinatorial optimization problems, e.g., plugging several $(1 + \lambda)$ -EAs within each subproblem. Finally, our strategies are inherently distributed in the sense that each sub-problem is optimized concurrently in parallel. In this respect, an interesting research issue would be to investigate the effective parallelization of our strategies.

References

1. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
2. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer (2007)
3. Wagner, T., Beume, N., Naujoks, B.: Pareto-, Aggregation-, and Indicator-based Methods in Many-objective Optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 742–756. Springer, Heidelberg (2007)

4. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE TEC* 11(6), 712–731 (2007)
5. Chiang, T.C., Lai, Y.P.: MOEA/D-AMS: Improving MOEA/D by an adaptive mating selection mechanism. In: CEC, pp. 1473–1480 (2011)
6. Ishibuchi, H., Sakane, Y., Tsukamoto, N., Nojima, Y.: Simultaneous Use of Different Scalarizing Functions in MOEA/D. In: GECCO, pp. 519–526. ACM (2010)
7. Hughes, E.J.: Multiple Single Objective Pareto Sampling. In: CEC, pp. 2678–2684 (2003)
8. Miettinen, K.: *Nonlinear Multiobjective Optimization*. Kluwer (1999)
9. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE TEC* 13(2), 284–302 (2009)
10. Giagkiozis, I., Purshouse, R.C., Fleming, P.J.: Generalized Decomposition. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 428–442. Springer, Heidelberg (2013)
11. Verel, S., Liefoghe, A., Jourdan, L., Dhaenens, C.: On the structure of multi-objective combinatorial search space: MNK-landscapes with correlated objectives. *EJOR* 227(2), 331–342 (2013)
12. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE TEC* 3(4), 257–271 (1999)
13. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, ETH Zurich (2006)
14. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE TEC* 7(2), 117–132 (2003)

MH-MOEA: A New Multi-Objective Evolutionary Algorithm Based on the Maximin Fitness Function and the Hypervolume Indicator

Adriana Menchaca-Mendez and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)

Departamento de Computación

México D.F. 07300, México

adriana.menchacamendez@gmail.com, ccoello@cs.cinvestav.mx

Abstract. In this paper, we propose an approach that combines a modified version of the maximin fitness function and the hypervolume indicator for selecting individuals into a Multi-Objective Evolutionary Algorithm (MOEA). Our proposed selection mechanism is incorporated into a MOEA which adopts the crossover and mutation operators of the Nondominated Sorting Genetic Algorithm-II (NSGA-II), giving rise to the so-called “Maximin-Hypervolume Multi-Objective Evolutionary Algorithm (MH-MOEA)”. Our proposed MH-MOEA is validated using standard test problems taken from the specialized literature, using from three to six objectives. Our results are compared with respect to those produced by MC-MOEA (which is based on the maximin fitness function and a clustering technique), MOEA/D using Penalty Boundary Intersection (PBI), which is based on decomposition and iSMS-EMOA (which is based on the hypervolume indicator). Our preliminary results indicate that our proposed MH-MOEA is a good alternative to solve multi-objective optimization problems having both low dimensionality and high dimensionality in objective function space.

1 Introduction

In the real world, there are many optimization problems which involve multiple objective functions (normally in conflict with each other) that need to be satisfied at the same time. They are called *multi-objective optimization problems (MOPs)*. In MOPs, the notion of optimality refers to the best possible trade-offs among all the objectives. Consequently, there are several optimal solutions (the so-called *Pareto optimal set* whose image is called the *Pareto front*). The use of evolutionary algorithms for solving MOPs has become very popular, giving rise to the so-called Multi-Objective Evolutionary Algorithms (MOEAs). We can classify MOEAs, based on their selection mechanism, into two groups: (i) those that incorporate the concept of Pareto optimality, and (ii) those that do not use Pareto dominance to select individuals. Since Pareto-based MOEAs have several limitations (from which the main one is that their behavior quickly degrades as

we increase the number of objectives), MOEAs of type (ii) have gained increasing popularity in the last few years.

We are interested in the *maximin fitness function* (MFF) and the *hypervolume indicator* (I_H). Both are of type (ii). MFF has some interesting properties and its complexity is linear with respect to the number of objective functions. I_H is the only unary indicator which is known to be “Pareto compliant” [18]. The main disadvantage of MOEAs based on I_H is their high computational cost. In this paper, we propose a hybrid of MFF and I_H for selecting individuals into a MOEA. The motivation behind this proposal is to alleviate the disadvantages of MFF which does not select well-distributed individuals [13,15]. Our conjecture is that it is possible to improve the approximation of the Pareto optimal set obtained by a MOEA based on MFF, if we can improve the diversity of the population at each generation. Therefore, we propose to use I_H to correct the possible errors produced when selecting with MFF. Finally, we incorporate our new selection mechanism into a MOEA that uses the crossover and mutation operators of NSGA-II to create new individuals. Our proposed MOEA is called “*Maximin-Hypervolume Multi-Objective Evolutionary Algorithm (MH-MOEA)*”.

The remainder of this paper is organized as follows. Section 2 states the problem of our interest. The maximin fitness function is described in Section 3. Section 4 describes the hypervolume indicator. Our proposal is discussed in Section 5. Our experimental validation and the results obtained are shown in Section 6. Finally, we provide our conclusions and some possible paths for future work in Section 7.

2 Problem Statement

We are interested in the general *multi-objective optimization problem (MOP)*, which is defined as follows: Find $\mathbf{x}^* = [x_1^*, x_2^*, \dots, x_n^*]^T$ which optimizes

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})]^T \quad (1)$$

such that $\mathbf{x}^* \in \Omega$, where $\Omega \subset R^n$ defines the feasible region of the problem. Assuming minimization problems, we have the following definitions.

Definition 1. We say that a vector $\mathbf{x} = [x_1, \dots, x_n]^T$ dominates vector $\mathbf{y} = [y_1, \dots, y_n]^T$, denoted by $\mathbf{x} \prec \mathbf{y}$, if and only if $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$ for all $i \in \{1, \dots, k\}$ and there exists an $i \in \{1, \dots, k\}$ such that $f_i(\mathbf{x}) < f_i(\mathbf{y})$.

Definition 2. We say that a vector $\mathbf{x} = [x_1, \dots, x_n]^T$ weakly dominates vector $\mathbf{y} = [y_1, \dots, y_n]^T$, denoted by $\mathbf{x} \preceq \mathbf{y}$, if \mathbf{x} is not worse than \mathbf{y} in all objectives.

Definition 3. A point $\mathbf{x}^* \in \Omega$ is Pareto optimal if there does not exist any $\mathbf{x} \in \Omega$ such that $\mathbf{x} \prec \mathbf{x}^*$.

Definition 4. For a given MOP, $\mathbf{f}(\mathbf{x})$, the Pareto optimal set is defined as: $\mathcal{P}^* = \{\mathbf{x} \in \Omega \mid \neg \exists \mathbf{y} \in \Omega : \mathbf{f}(\mathbf{y}) \prec \mathbf{f}(\mathbf{x})\}$.

Definition 5. Let $\mathbf{f}(\mathbf{x})$ be a given MOP and \mathcal{P}^* the Pareto optimal set. Then, the Pareto Front is defined as: $\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\}$.

3 Maximin Fitness Function

The maximin fitness function (MFF) was proposed by Balling [3]. Balling and Wilson proposed the modified MFF [5] which works as follows. Let's consider a MOP with K objective functions. Let f_k^i be the normalized value of the k^{th} objective for the i^{th} individual in a particular generation. Then, the modified MFF of individual i is defined as:

$$fitness^i = max_{j \neq i, j \in \mathcal{ND}} (min_k (f_k^i - f_k^j)) \tag{2}$$

where \mathcal{ND} is the set of non-dominated individuals. The min is taken over all the objective functions, and the max is taken over all non-dominated individuals in the population, except for the same individual i . From eq. (2), we can say the following: any individual whose maximin fitness is greater than zero is a dominated individual, any individual whose maximin fitness is less than zero is a non-dominated individual and, any individual whose maximin fitness is equal to zero is a weakly nondominated individual. Also, MFF penalizes clustering of non-dominated individuals and the maximin fitness of dominated individuals is a metric of the distance to the non-dominated front.

MFF and its modified version have been incorporated into evolutionary algorithms such as genetic algorithms [5,4], particle swarm [11,12] and ant colony [10] optimizers. However, in those papers, only low dimensionality MOPs were considered and no extra diversity mechanism was adopted based on the idea that MFF penalizes clustering. In recent years, two important disadvantages of MFF were identified in [13]. The main disadvantage of MFF is related to the following question: **Is it better to prefer weakly nondominated individuals to dominated individuals?** The answer provided in [13] was that it is not good to prefer weakly nondominated individuals (even if they are weakly nondominated by any dominated individual). The authors showed in [13,15] that if we use a MOEA based on MFF to solve a MOP in which one objective function is easier to solve than the others, it is likely that the MOEA only obtains weakly Pareto points or that its convergence slows down. In order to address this problem, the following constraint was proposed in [13]: *Any individual that we want to select must not be similar (in any objective function) to another (selected) individual.*

The second disadvantage of MFF has to do with the poor diversity obtained in objective function space when we use it to select individuals. In [13,15], the authors proposed to combine either MFF or its modified version with a clustering technique in order to improve diversity.

4 Hypervolume Indicator

The hypervolume indicator (I_H) was originally proposed by Zitzler and Thiele in [17]. If Λ denotes the Lebesgue measure, I_H is defined as:

$$I_H(\mathcal{A}, \mathbf{y}_{ref}) = \Lambda \left(\bigcup_{\mathbf{y} \in \mathcal{A}} \{\mathbf{y}' \mid \mathbf{y} \prec \mathbf{y}' \prec \mathbf{y}_{ref}\} \right) \tag{3}$$

where $\mathbf{y}_{ref} \in \mathbb{R}^k$ denotes a reference point that should be dominated by all the Pareto optimal points. The contribution to I_H of a solution \mathbf{x} is defined as:

$$C_H(\mathbf{x}, \mathcal{A}) = I_H(\mathcal{A}, \mathbf{y}_{ref}) - I_H(\mathcal{A} \setminus \mathbf{x}, \mathbf{y}_{ref}) \quad (4)$$

where $\mathbf{x} \in \mathcal{A}$. Then, the contribution of \mathbf{x} is the space that is only covered by \mathbf{x} .

Perhaps, the most popular MOEA based on I_H is the S metric selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) [8]. SMS-EMOA generates only one solution by iteration using the crossover and mutation operators of the NSGA-II. After that, it applies Pareto ranking. When the last front has more than one solution, SMS-EMOA uses the contribution to I_H to decide which solution will be removed. Therefore, when all individuals are non-dominated, SMS-EMOA needs to calculate the contribution to I_H of all individuals in the population and the contribution of the new individual. This is not good because we know that calculating these contributions is computationally expensive for more than three objective functions. There are other MOEAs based on I_H . However, most of them use the same competition scheme. Recently, a new selection scheme based on I_H and its locality property [2,1] was proposed in [14]. It works as follows: Let's assume that at each iteration of a MOEA, only one solution \mathbf{x}_{new} is created and the current population is \mathcal{P} . Then, we choose the nearest neighbor (\mathbf{x}_{near}) of \mathbf{x}_{new} in \mathcal{P} and we also choose (randomly) another solution, \mathbf{x}_{rand} , such that $\mathbf{x}_{rand} \in \mathcal{P}$ and $\mathbf{x}_{rand} \neq \mathbf{x}_{near}$. After that, \mathbf{x}_{rand} , \mathbf{x}_{new} and \mathbf{x}_{near} will compete to survive. The solution with the worst contribution to I_H is eliminated.

5 Our Proposed Approach

We propose here a selection mechanism based on the modified MFF and I_H . The idea is to use the modified MFF as our main selection mechanism and I_H to correct its possible errors. Unlike SMS-EMOA [8] or iSMS-EMOA [14], in which only one individual is created per iteration, our mechanism is designed to work with population schemes. This is possible for two reasons: The maximin fitness of each individual determines the order in which each individual competes to survive using I_H and in the competition scheme proposed in [14] each individual only competes with two other individuals of the population (its nearest neighbor and a randomly selected individual). Then, the combinatorial problem no longer exists.

Our selection mechanism works as follows: If we want to select S individuals from a population \mathcal{P} , we assign first a fitness value to each individual using the modified MFF (see eq. (2)). Then, we proceed to select the individuals according to their fitness, verifying similarity between selected individuals (see Algorithm 1, lines 5 to 11). If we consider all individuals in the population and we do not select S individuals, we select the remaining individuals considering only the maximin fitness (see Algorithm 1, lines 13 to 20). If we already selected the S individuals but there are still non-dominated individuals in \mathcal{P} who have not participated in the selection process, then, we proceed to use the contribution to I_H as follows: Let \mathcal{S} be the set of current selected individuals. Then, for each

nondominated individual $\mathcal{P}[i]$ who has not participated in the selection process, we obtain the index of its nearest neighbor in \mathcal{S} (we call it NN) and we choose a random index RI such that $RI \in \{1, \dots, |\mathcal{S}|\}$ and $RI \neq NN$. Finally, we calculate the contribution to I_H of $\mathcal{P}[i]$, $\mathcal{S}[NN]$ and $\mathcal{S}[RI]$. If $\mathcal{P}[i]$ has a better contribution than $\mathcal{S}[NN]$ or $\mathcal{S}[RI]$, then $\mathcal{P}[i]$ replaces the individual with the worst contribution ($\mathcal{S}[NN]$ or $\mathcal{S}[RI]$). See Algorithm 1, lines 22 to 34). The process to verify similarity between individuals is shown in Algorithm 2, where min_dif is the minimum difference allowed between solutions with respect to all objective functions and K is the number of objective functions.

In order to evaluate our new selection mechanism, we incorporate it into a MOEA that uses the crossover and mutation operators of NSGA-II to create new individuals. The proposed MOEA is called “**Maximin-Hypervolume Multi-Objective Evolutionary Algorithm (MH-MOEA)**” and it works as follows: If the size of the population is P , then we create P new individuals. We use a binary tournament to select the parents. At each tournament, two individuals are randomly selected and the one with the higher maximin fitness value is chosen. After that, we combine the population of parents and offspring to obtain a population of size $2P$. Then, we use our proposed selection mechanism to choose the P individuals that will take part of the following generation. This process is repeated for a certain (pre-defined) number of generations.

6 Experimental Results

We compared our proposed MH-MOEA with respect to MC-MOEA [15] (the version in which the modified MFF is used all the time), MOEA/D [16] (using PBI to decompose the MOP¹) and iSMS-EMOA [14].² For MOEA/D, we generated the convex weights using the technique proposed in [6] and after that, we applied clustering (k -means) to obtain a specific number of weights. It is worth noticing that all of these MOEAs use the same operators to create new individuals, which allows a fair comparison of the selection operators.

For our experiments, we adopted seven problems from the Deb-Thiele-Lauermanns-Zitzler (DTLZ) test suite [7]. We used $k = 5$ for DTLZ1, DTLZ3 and DTLZ6 and $k = 10$ for the remaining test problems. Also, we adopted seven problems from the Walking-Fish Group (WFG) toolkit [9], with $k_factor = 2$ and $l_factor = 10$. For each test problem, we performed 30 independent runs. For all algorithms, we adopted the parameters suggested by the authors of NSGA-II: $p_c = 0.9$ (crossover probability), $p_m = 1/n$ (mutation probability), where n is the number of decision variables. Both for the crossover and mutation operators, we adopted $\eta_c = 15$ and $\eta_m = 20$, respectively. In the case of MC-MOEA and our MH-MOEA, we used $min_dif = 0.0001$. Regarding to MOEA/D, we used a

¹ We decided to use the PBI approach because the resultant optimal solutions in the PBI should have a more uniform distribution than those obtained by the Tchebycheff approach [16].

² The source code of the all algorithms used here can be provided by the first author upon request.

Algorithm 1. Maximin-Hypervolume Selection

```

Input :  $\mathcal{P}$  (Population),  $S$  (number of individuals to choose  $S < \|\mathcal{P}\|$ ).
Output:  $\mathcal{S}$  (Selected individuals).
/*Assign fitness to each individual in the population, using the modified maximin
fitness function */
1 AssignFitness( $\mathcal{P}$ );
2  $numNonDom \leftarrow$  Number of nondominated solutions in  $\mathcal{P}$ ;
/*Sorting with respect to the maximin fitness */
3 Sort( $\mathcal{P}$ );
4  $s \leftarrow 1, i \leftarrow 1, \mathcal{S} \leftarrow \emptyset$ ;
/*Fill up the new population with the best copies according to the maximin fitness,
verifying that there is not a similar one */
5 while  $s \leq S$  AND  $i \leq \|\mathcal{P}\|$  do
6   if  $\mathcal{P}[i]$  is not similar to any individual in  $\mathcal{S}$  then
7     if  $\mathcal{P}[i]$  is not similar to any individual in  $\mathcal{S}$  then
8        $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{P}[i]$ ;
9        $s \leftarrow s + 1$ ;
10    end
11    $i \leftarrow i + 1$ ;
12 end
13 if  $s \leq S$  then
14   /*Choose the remaining individuals considering only the maximin fitness */
15    $i \leftarrow 1$ ;
16   while  $s \leq S$  do
17     if  $\mathcal{P}[i]$  has not been selected then
18        $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{P}[i]$ ;
19        $s \leftarrow s + 1$ ;
20     end
21      $i \leftarrow i + 1$ ;
22   end
23 else
24   /*Improve the diversity according to the contribution to  $I_H$  */
25   while  $i < numNonDom$  do
26     if  $\mathcal{P}[i]$  is not similar to any individual in  $\mathcal{S}$  then
27        $NN \leftarrow$  Index of nearest neighbor to  $\mathcal{P}[i]$  in  $\mathcal{S}$ ;
28        $RI \leftarrow$  Index of a randomly selected individual in  $\mathcal{S}$  such that  $NN \neq RI$ ;
29       /*Calculate the contributions to the hypervolume */
30        $C_{NN} \leftarrow C_H(\mathcal{S}[NN], \mathcal{S})$ ;
31        $C_{RI} \leftarrow C_H(\mathcal{S}[RI], \mathcal{S})$ ;
32        $C_i \leftarrow C_H(\mathcal{P}[i], \mathcal{S})$ ;
33       /*Remove the individual with the worst contribution */
34        $worst \leftarrow$  Index of the individual with the worst contribution ( $NN, RI$  or  $i$ );
35       if  $worst = NN$  or  $worst = RI$  then
36         Replace  $\mathcal{S}[worst]$  with  $\mathcal{P}[i]$ ;
37       end
38     end
39   end
40 end
41 return  $\mathcal{S}$ ;

```

Algorithm 2. Verify similarity

```

Input :  $x$  (individual),  $\mathcal{S}$  (population).
Output: Returns 1, if the individual  $x$  is similar to any individual in the population  $\mathcal{S}$ ;
otherwise, returns 0.
1 for  $i \leftarrow 1$  to  $\|\mathcal{S}\|$  do
2   for  $k \leftarrow 1$  to  $K$  do
3     if  $|x.f[k] - \mathcal{S}[i].f[k]| < min\_dif$  then
4       return 1;
5     end
6   end
7 end
8 return 0;

```

niche of size 20. We performed a maximum of 50,000 fitness function evaluations (we used a population size of 100 individuals and we iterated for 500 generations). Only in DTLZ3 we performed 100,000 evaluations (we used a population size of 100 individuals and we iterated for 1000 generations).

6.1 Performance Indicators

We adopted only I_H to validate our results because it rewards both convergence towards the Pareto front as well as the maximum spread of the solutions obtained. Also, I_H is Pareto compliant. To calculate the hypervolume indicator, we used the following reference points: $y_{ref} = [y_1, \dots, y_M]$ such that $y_i = 0.7$ for DTLZ1, $y_{ref} = [y_1, \dots, y_M]$ such that $y_i = 1.1$ for DTLZ(2-6), $y_{ref} = [y_1, \dots, y_M]$ such that $y_M = 6.1$ and $y_{i \neq M} = 1.1$ for DTLZ7. In the case of the WFG test problems, we generated the reference point using a value slightly higher than the highest value found for each objective function taking into account all the outputs of the algorithms.

6.2 Discussion of Results

In Table 1, we present the results obtained with respect to I_H as well as the statistical analysis applied to the experiments using Wilcoxon's rank sum. In (a), we can see that our MH-MOEA obtained better results than MC-MOEA in eleven problems. It is important to see that we can reject the null hypothesis "medians are equal" in all cases, and then, we can say that in these problems our MH-MOEA outperformed MC-MOEA. Only in DTLZ1, MC-MOEA outperformed our MH-MOEA. In (b), we compare our MH-MOEA with respect to MOEA/D and we can see that it outperformed MOEA/D in eleven cases and, only in DTLZ1, MOEA/D outperformed our MH-MOEA. Finally, in (c), we can see that iSMS-EMOA outperformed our MH-MOEA in four problems, our MH-MOEA outperformed iSMS-EMOA in four problems and, in four problems, we can observe that the null hypothesis cannot be rejected, which means that both algorithms have a similar behavior. Also, in (d), (e) and (f), we show a scalability analysis with respect to the number of objectives for some of the problems adopted, using four, five and six objective functions. In these Tables, we can see that our MH-MOEA continues to work well when we increase the number of objectives. For example, we can say that our MH-MOEA outperformed MC-MOEA and MOEA/D because it obtained better results with respect to I_H , in all problems, and only in two cases, the null hypothesis cannot be rejected: In DTLZ3 with six objectives, our MH-MOEA has a behavior similar to MC-MOEA and in DTLZ7 with six objective functions, our MH-MOEA has a similar behavior to MOEA/D. With respect to iSMS-EMOA, our MH-MOEA outperformed iSMS-EMOA in two problems, it is outperformed by iSMS-EMOA in four problems and they have a similar behavior in three problems. Finally, in Table 2, we present plots of the average running time required by each algorithm to find the approximation of the Pareto optimal set in the problems adopted for the scalability analysis and we can note that MOEA/D and MC-MOEA are the

fastest algorithms. However, in Table 1, we saw that they are outperformed by our proposed MH-MOEA. An interesting thing is that our MH-MOEA requires less running time than iSMS-EMOA and, as we saw in Table 1, it obtains competitive results with respect to iSMS-EMOA.

Table 1. Comparison of results in the DTLZ and WFG test problems using I_H . We show average values over 30 independent runs. The values in parentheses correspond to the standard deviations. The third column of each table shows the results of the statistical analysis applied to our experiments using Wilcoxon's rank sum. $H = 0$ indicates that the null hypothesis ("medians are equal") cannot be rejected at the 5% level. $H = 1$ indicates that the null hypothesis can be rejected at the 5% level.

f	mc-moea I_H	mh-moea I_H	H	moead I_H	mh-moea I_H	H	isms-emoa I_H	mh-moea I_H	H
DTLZ1(3)	0.311(0.00)	0.301(0.05)	1	0.303(0.00)	0.301(0.05)	1	0.206(0.09)	0.301(0.05)	1
DTLZ2(3)	0.696(0.00)	0.757(0.00)	1	0.708(0.00)	0.757(0.00)	1	0.757(0.00)	0.757(0.00)	1
DTLZ3(3)	0.666(0.02)	0.732(0.07)	1	0.702(0.00)	0.732(0.07)	1	0.049(0.12)	0.732(0.07)	1
DTLZ5(3)	0.424(0.00)	0.439(0.00)	1	0.416(0.00)	0.439(0.00)	1	0.439(0.00)	0.439(0.00)	1
DTLZ7(3)	1.851(0.20)	1.939(0.21)	1	1.607(0.20)	1.939(0.21)	1	1.923(0.22)	1.939(0.21)	0
WFG1(3)	17.62(1.37)	21.03(0.60)	1	16.21(0.31)	21.03(0.60)	1	21.21(0.16)	21.03(0.60)	0
WFG2(3)	0.115(0.00)	0.118(0.01)	1	0.088(0.00)	0.118(0.01)	1	0.124(0.00)	0.118(0.01)	1
WFG3(3)	0.407(0.00)	0.465(0.00)	1	0.388(0.01)	0.465(0.00)	1	0.466(0.00)	0.465(0.00)	1
WFG4(3)	20.85(0.55)	33.80(0.33)	1	22.85(0.54)	33.80(0.33)	1	29.38(0.07)	33.80(0.33)	1
WFG5(3)	7.997(0.20)	9.877(0.01)	1	8.501(0.14)	9.877(0.01)	1	9.875(0.01)	9.877(0.01)	0
WFG6(3)	0.920(0.01)	1.024(0.00)	1	0.845(0.00)	1.024(0.00)	1	1.023(0.00)	1.024(0.00)	0
WFG7(3)	17.28(0.72)	22.74(0.08)	1	16.22(1.85)	22.74(0.08)	1	23.89(0.10)	22.74(0.08)	1

(a)

(b)

(c)

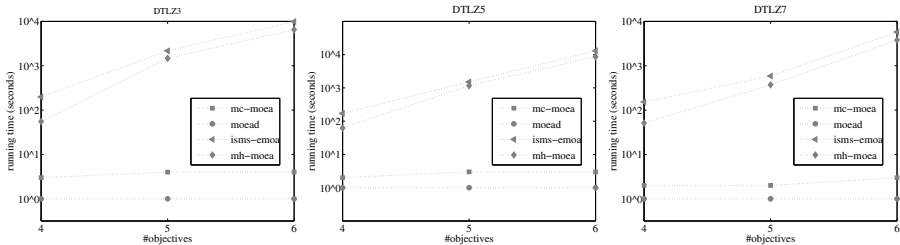
f	mc-moea I_H	mh-moea I_H	H	moead I_H	mh-moea I_H	H	isms-emoa I_H	mh-moea I_H	H
DTLZ3(4)	0.857(0.03)	1.027(0.03)	1	0.849(0.00)	1.027(0.03)	1	1.013(0.10)	1.027(0.03)	1
DTLZ5(4)	0.205(0.02)	0.436(0.00)	1	0.395(0.00)	0.436(0.00)	1	0.437(0.00)	0.436(0.00)	1
DTLZ7(4)	0.554(0.06)	0.758(0.10)	1	0.510(0.13)	0.758(0.10)	1	0.739(0.16)	0.758(0.10)	0
DTLZ3(5)	0.999(0.04)	1.117(0.32)	1	0.907(0.01)	1.117(0.32)	1	1.288(0.02)	1.117(0.32)	1
DTLZ5(5)	0.164(0.02)	0.444(0.00)	1	0.384(0.00)	0.444(0.00)	1	0.446(0.00)	0.444(0.00)	1
DTLZ7(5)	0.064(0.01)	0.153(0.05)	1	0.090(0.02)	0.153(0.05)	1	0.164(0.05)	0.153(0.05)	0
DTLZ3(6)	1.060(0.09)	1.426(0.22)	1	0.836(0.12)	1.426(0.22)	1	1.1780(0.60)	1.426(0.22)	1
DTLZ5(6)	0.138(0.04)	0.453(0.00)	1	0.386(0.00)	0.453(0.00)	1	0.461(0.00)	0.453(0.00)	1
DTLZ7(6)	0.003(0.00)	0.020(0.01)	1	0.017(0.00)	0.020(0.01)	0	0.027(0.00)	0.020(0.01)	0

(d)

(e)

(f)

Table 2. Time required by MC-MOEA, MOEA/D, iSMS-EMOA and our proposed MH-MOEA for the test problems adopted in the scalability analysis. $s =$ seconds. All algorithms were compiled using the GNU C compiler and they were executed on a computer with a 2.66GHz processor and 4GB in RAM. We can see that the worst algorithm, regarding running time, is ISMS-EMOA in all three MOPs (DTLZ3, DTLZ5 and DTLZ7 with 4, 5 and 6 objective functions). While the best algorithm is MOEA/D. Also, we can see that our MH-MOEA outperforms iSMS-EMOA in all cases.



7 Conclusions and Future Work

We have proposed a new selection mechanism based on the modified maximin fitness function (MFF) and the hypervolume indicator (I_H). Unlike other selection mechanisms based on I_H , such as the one adopted in the SMS-EMOA algorithm or its improved version (iSMS-EMOA), our selection mechanism works with populations. Our idea is to use the modified MFF as our main selection mechanism and I_H to correct the possible errors in the selection process. Our preliminary results indicate that our MH-MOEA is able to outperform MOEAs such as MC-MOEA and MOEA/D, both with few and many objectives. Also, MH-MOEA is competitive with respect to iSMS-EMOA, outperforming it in some cases, while requiring a lower computational time.

As part of our future work, we are interested in studying mechanisms to approximate the contribution to I_H and to use one of them instead of adopting the exact calculation. Since we only use I_H to correct possible errors generated when we select with the maximin fitness, we expect to retain, as much as possible, the quality in our solutions when we approximate the contribution to I_H .

References

1. Auger, A., Bader, J., Brockhoff, D.: Theoretically Investigating Optimal μ -Distributions for the Hypervolume Indicator: First Results for Three Objectives. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 586–596. Springer, Heidelberg (2010)
2. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the Hypervolume Indicator: Optimal $\{\mu\}$ -Distributions and the Choice of the Reference Point. In: FOGA 2009: Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, Orlando, Florida, USA, pp. 87–102. ACM (January 2009)
3. Balling, R.: Pareto sets in decision-based design. *Journal of Engineering Valuation and Cost Analysis* 3, 189–198 (2000)
4. Balling, R.: The Maximin Fitness Function; Multiobjective City and Regional Planning. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 1–15. Springer, Heidelberg (2003)
5. Balling, R., Wilson, S.: The Maximin Fitness Function for Multi-objective Evolutionary Computation: Application to City Planning. In: Spector, L., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001, pp. 1079–1084. Morgan Kaufmann Publishers, San Francisco (2001)
6. Das, I., Dennis, J.E.: Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM J. on Optimization* 8(3), 631–657 (1998)
7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, USA (2005)
8. Emmerich, M.T.M., Beume, N., Naujoks, B.: An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)

9. Huband, S., Hingston, P., Barone, L., While, L.: A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation* 10(5), 477–506 (2006)
10. Li, H., Huang, X., Feng, Q.: Optimizing expressway maintenance planning by coupling ant algorithm and geography information system transportation in hubei province, china. In: 2011 IEEE International Geoscience and Remote Sensing Symposium, IGARSS, pp. 2977–2979 (July 2011)
11. Li, X.: Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 117–128. Springer, Heidelberg (2004)
12. Li, X., Branke, J., Kirley, M.: On Performance Metrics and Particle Swarm Methods for Dynamic Multiobjective Optimization Problems. In: 2007 IEEE Congress on Evolutionary Computation, CEC 2007, pp. 576–583. IEEE Press (September 2007)
13. Menchaca-Mendez, A., Coello Coello, C.A.: Solving Multi-Objective Optimization Problems using Differential Evolution and a Maximin Selection Criterion. In: 2012 IEEE Congress on Evolutionary Computation, CEC 2012, Brisbane, Australia, June 10–15, pp. 3143–3150. IEEE Press (2012)
14. Menchaca-Mendez, A., Coello Coello, C.A.: A New Selection Mechanism Based on Hypervolume and its Locality Property. In: 2013 IEEE Congress on Evolutionary Computation, CEC 2013, Cancún, México, June 20–23, pp. 924–931. IEEE Press (2013)
15. Menchaca-Mendez, A., Coello Coello, C.A.: Selection Operators based on Maximin Fitness Function for Multi-Objective Evolutionary Algorithms. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 215–229. Springer, Heidelberg (2013)
16. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11(6), 712–731 (2007)
17. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN V. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation* 7(2), 117–132 (2003)

Empirical Performance of the Approximation of the Least Hypervolume Contributor

Krzysztof Nowak¹, Marcus Mörtens², and Dario Izzo¹

¹ European Space Agency, Noordwijk, The Netherlands

² TU Delft, Delft, The Netherlands

Abstract. A fast computation of the hypervolume has become a crucial component for the quality assessment and the performance of modern multi-objective evolutionary optimization algorithms. Albeit recent improvements, exact computation becomes quickly infeasible if the optimization problems scale in their number of objectives or size. To overcome this issue, we investigate the potential of using approximation instead of exact computation by benchmarking the state of the art hypervolume algorithms for different geometries, dimensionality and number of points. Our experiments outline the threshold at which exact computation starts to become infeasible, but approximation still applies, highlighting the major factors that influence its performance.

Keywords: Hypervolume indicator, performance indicators, multi-objective optimization, many-objective optimization, approximation algorithms.

1 Introduction

The *hypervolume indicator* (also known as Lebesgue measure [1] or S-metric [2]) is a popular quality measure for multi-objective optimization [3]. It was first suggested by Zitzler and Thiele [4] as the size of the objective space covered by the non-dominated solutions. It has the property of being strictly Pareto-compliant, i.e., the Pareto-optimal front guarantees the maximum possible hypervolume while any dominated set will assign a lower hypervolume [5]. Besides its application as a performance indicator, the exclusive contribution of one individual to the total hypervolume is used by Multi-Objective Optimization Algorithms (MOEA) for selection, diversification and archiving. Because well-established Pareto dominance-based MOEA like NSGA-II [6] and SPEA2 [7] deteriorate in performance as the number of objectives increases, indicator-based algorithms [8] such as SMS-EMOA [9], HypE [10] and MO-CMAES [11] provide an alternative optimization design. A fast computation of the hypervolume indicator and the contributions has thus become important for multi-objective optimization.

Bringmann and Friedrich [12] show that computing the hypervolume indicator is $\#\mathcal{P}$ -complete, which implies that there exists no polynomial-time algorithm unless $\mathcal{P} = \mathcal{NP}$. Similar hardness results hold for computing hypervolume contributions. However, the same authors also show that computing the hypervolume

is fixed parameter tractable in the average case which gives hope that exact algorithms might be useful in practice [13]. Given this opportunity, we investigate and compare the state of the art approaches for determining the least contributor: exact algorithms approaching the problem from a computational geometry standpoint and the approximation algorithm by Bringmann and Friedrich [12] which can offer a better runtime at the cost of the precision.

2 Related Work

For dimensions $d \leq 4$, hypervolume computation is in general tractable, as there exist algorithms tailored for $d = 2, 3$ and 4 [14–16]. For arbitrary dimensions, the best currently known algorithms in terms of asymptotical runtime all rely on the *Klee measure problem*, of which the computation of the hypervolume indicator is a special case. For $d \geq 7$ Bringmann [17] gives an algorithm that runs in $\mathcal{O}(n^{\frac{d+2}{3}})$ whereas Yildiz and Suri [18] are asymptotically better for $d = 4, 5, 6$ with $\mathcal{O}(n^{\frac{d-1}{2}} \log n)$. The HOY [19] algorithm uses a space partitioning based on the divide and conquer paradigm and runs in $\mathcal{O}(n^{\frac{d}{2}} \log n)$. The drawback of these methods are the large data structures that need to be maintained during the run. The fastest algorithm based on dimension-sweeping is the FPL algorithm by Fonseca et al. [20] that has an asymptotic complexity of $\mathcal{O}(n^{d-2} \log n)$ while using only linear space. The WFG [21] is based on using bounding boxes to determine the exclusive contributions of points, which are then used to compute the total hypervolume. Despite its asymptotic run-time of $\mathcal{O}(n^d)$, there is experimental [22] and theoretical evidence [13] that it is currently the fastest exact algorithm for higher dimensions in practice. Although a good average case complexity was proven, hypervolume computations are still tremendously time-consuming for today's computers already starting with $d = 10$. Approximation algorithms and heuristics make it possible to explore problem domains of $d \geq 10$, which often needed to be scaled down (i.e. by aggregation) before to become tractable. There exists a fully polynomial randomized approximation scheme (FPRAS) for the hypervolume indicator [23], which allows for its approximation with given precision and probability in polynomial time. Ishibuchi et al. [24] give a heuristic that uses achievement scalarizing functions to approximate the hypervolume indicator, however, no approximation ratio is known. Essential topic of our research is the algorithm of Bringmann and Friedrich [12] which combines a Monte Carlo-like sampling method with a racing approach to directly approximate the least hypervolume contributor. It is the only algorithm we are aware of that gives a guarantee that for any given $\delta, \varepsilon \geq 0$ the obtained solution is with a probability of $(1 - \delta)$ larger by at most a factor of $(1 + \varepsilon)$ than the least contributor.

Closely related to this work is the one of Bringmann et. al [25] where the empirical performance of MO-CMAES is evaluated when using the approximation algorithm as a subroutine. While it was shown that the runtime of MO-CMAES could be reduced, it is unclear which sort of geometries during evolution had to be processed by the approximation algorithm and how fast it did so.

Also, the problem with highest dimensionality chosen was $d = 8$, for which arguably an exact computation might still be feasible. As we will analyze the runtime of the approximation algorithm unbiased by any MOEA from 2 to 100 dimensions, our approach is more direct and comprehensive, extending the preliminary results from the original work. In particular, we follow the suggestion made by the authors at the end of the original work [12] to create a broader experimental setup, in which we can observe the occurrence and influence of hard cases along other factors that impact the runtime by orders of magnitude, but were never addressed in previous works before.

3 Experimental Setup

In this section we describe the datasets we used for our experiments¹. Since the hypervolume indicator computation is inherently connected with a reference point, we propose a robust selection procedure. Finally, we assess the set of the best-performing exact algorithms to provide a reliable comparison with the approximation algorithm in the next section.

3.1 Datasets

We assume minimization in our setup and that each point in the dataset is constrained to a unit box $[0, 1]^d$. We generate the datasets with three analytically-defined geometries (*Convex*, *Concave* and *Planar*). In order to show the performance of the algorithms on the maximization problems of the same geometries (and also to close the gap between the publications that assume it), we provide their inverted variants: *InvertedConvex*, *InvertedConcave* and *InvertedPlanar*.

Datasets were generated by sampling a multivariate normal distribution, similarly to how it was outlined in [12]. Let $q = (q_1, q_2, \dots, q_d)$ be a vector of d normal deviates, and $p \in \mathbb{R}^+$ a parameter of the norm in L^p space. We sample n non-dominated points using S :

$$S(q, p) = \frac{(|q_1|, |q_2|, \dots, |q_d|)}{(q_1^p + q_2^p + \dots + q_d^p)^{\frac{1}{p}}}. \quad (1)$$

For $p = 2$, $p = 1$ and $p = 0.5$ we obtain the *Convex*, *Planar* and *Concave* shapes respectively. Inverted variants of the shapes above involve the extra step of multiplication of each obtained vector by a scalar of -1 and translation by the vector $(1, 1, \dots, 1)$:

$$S_{Inverted}(q, p) = (1, 1, \dots, 1) - S(q, p). \quad (2)$$

Figure 1 visualizes the shapes for $d = 3$.

Additionally, we generate a group of *Random* datasets. *Random* dataset A (initially empty) is obtained by a repeated sampling of a point inside the box $[0, 1]^d$ and performing a test for the pairwise non-dominance with all of the previously sampled points in A . This process continues until $|A| = n$.

¹ Source code available: <https://github.com/esa/pagmo/wiki/Hypervolume>.

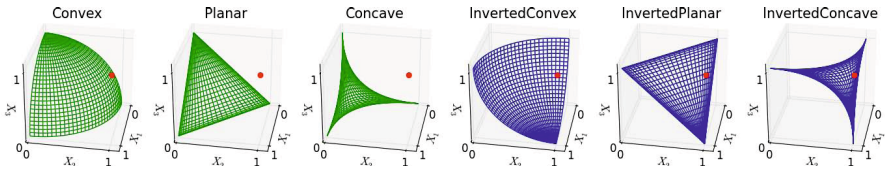


Fig. 1. Regular (green) and inverted (blue) dataset shapes, possible reference point (red) at (1, 1, 1)

Dataset size ranges from 10 to 100 points by a step of 10 and then from 100 to a 1000 points by a step of 100. Dimension ranges from 2 to 20 by a step of one, and then from 20 to 100 by a step of 10. We generate a sample of 10 datasets in each shape and for each combination of dimension and number of points. Although current state of the art in multi-objective optimization does not deal with problems of such extreme dimensionality, we are interested in presenting the empirical scaling capabilities of the algorithm itself.

3.2 Reference Point

Every hypervolume computation requires a reference point. One way of obtaining it, is simply assuming a fixed point which is guaranteed to be strongly dominated by all of the points in the set. When the hypervolume is employed for the optimization scenarios, this information might not be known upfront. In such cases, the reference point is chosen dynamically, i.e. dependent on the point set. A common approach is constructing a point out of the maxima in each coordinate (known as the *nadir point*), and offsetting it by a small constant. This assures that the points on the boundaries of the space have non-zero contribution to the total hypervolume, as it was explained in the work of Beume et al. [9]. However, a constant offset for the reference point may lead to problems, as any fixed value may be relatively large in comparison to the space boundary. In such case, border points may be influencing the hypervolume too strongly. In order to avoid that, we will shift the nadir point relatively to the boundary of the point set, as it was proposed by Knowles [26]. With N as the nadir point, and I as the ideal point (minima among all coordinates), we compute the reference point as follows:

$$R = N + \alpha \cdot (N - I). \tag{3}$$

For our experiment we assume $\alpha = 0.01$, resulting in a reference point shifted by 1% in each of the dimensions of the bounding box.

3.3 Selecting the Exact Algorithms and Experiment Design

We test selected exact hypervolume algorithms in order to determine a robust and efficient candidate, which we will use for the comparison with the approximation algorithm. We consider three dimension-specific algorithms: Dimension-sweep algorithm for $d = 2$ (HV2D), algorithm by Beume for $d = 3$ (HV3D)

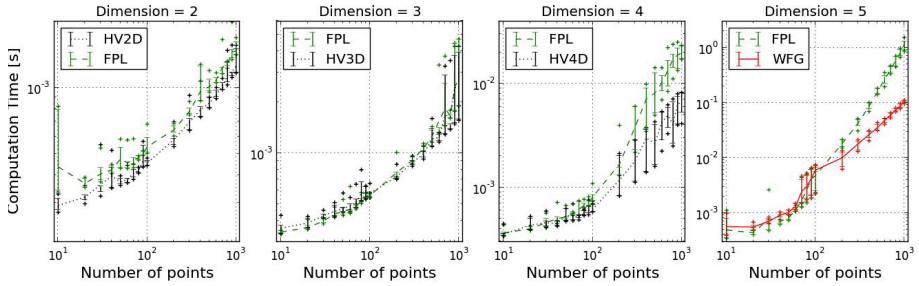


Fig. 2. Distribution of 10 hypervolume indicator computation times (*Random* dataset)

and algorithm by Guerreiro for $d = 4$ (HV4D). Besides these, we test three dimension-independent algorithms: WFG, FPL and HOY. All of the implementations of algorithms used for this research, can be found in the PaGMO library².

Figure 2 shows the median times of 10 hypervolume indicator computations for algorithms WFG, FPL, and the family of dimension-specific algorithms: HV2D, HV3D and HV4D, with whiskers describing the middle 8 runtimes (outliers are denoted with a “plus” mark). On each figure only the best and the second best performing algorithms are displayed per dimension. All of the dimension-specific algorithms tend to perform no worse in their domain than the dimension-independent ones. Out of the dimension-independent algorithms, WFG performs better than FPL and HOY on every test instance with more than 5 dimensions. For 5 dimensions, FPL performs better than WFG for 80 points or less.

Least contributor is obtained through $n + 1$ computations of the hypervolume:

$$\text{LeastContributor}(S) = \underset{p \in S}{\operatorname{argmin}}(\text{Hypervolume}(S) - \text{Hypervolume}(S \setminus \{p\})). \quad (4)$$

We improve on that by employing the algorithm by Emmerich et al. [16] for 3 dimensions, and a version of WFG optimized for the least contributor computation. Figure 3 presents the results for the least contributor computation. For $d \geq 4$ WFG outperformed all other algorithms.

We propose a group of best performing algorithms for the computation of the least contributor, which we will compare with the PaGMO implementation of the approximation algorithm by Bringmann and Friedrich [12] (to which we will refer as BFA). For that task we select all dimension-specific algorithms for $d \leq 3$ and WFG in every other case. Because BFA employs a mechanism in which difficult subproblems can be solved using the computation of the hypervolume indicator, we define a set of the best performing algorithms for the hypervolume indicator as well. For $d \leq 4$ we will use all of the dimension-specific algorithms, for $d = 5$ and $n < 80$ we will employ FPL, while the remaining cases will be handled by the WFG algorithm. We run the approximation algorithm with the parameters recommended by the authors, namely $\delta = 10^{-6}$ and $\varepsilon = 10^{-2}$. All experiments

² Available online at <https://github.com/esa/pagmo>

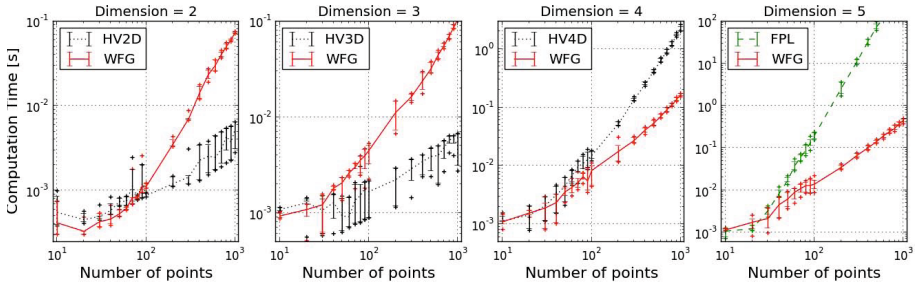


Fig. 3. Distribution of 10 least hypervolume contributor computation times (*Random* dataset)

were conducted on an Intel(R) Xeon(R) CPU E5-2650 @ 2.00GHz with 20480 KB of cache.

4 Results

This sections covers the results obtained by our comparison experiment. As exact computation of the least contributor is intractable for the majority of our test cases – rendering the measurement of empirical accuracy of BFA infeasible – we have to restrict our analysis to the runtime only.

Runtime patterns derived for the exact algorithms and the BFA algorithm can be seen in Figure 4. Given the extent of the data, we terminate each computation after 30 seconds to make the experiment feasible and to show the general outlook of the runtime patterns. First, we observe that *Planar*, *Concave* and *Convex* shapes were similar in the runtime patterns they produced, thus we used *Convex* as a representative of this group of shapes. For the same reason *InvertedConvex* was chosen as the representative of the inverted variants of the shapes above. The runtime pattern of the exact algorithm was similar across all shapes, while BFA performed worse for the first group (*Planar*, *Concave* and *Convex*) when compared with their corresponding inverted variants. Figures 4(c), 4(d) and Figures 4(e), 4(f) show a significant difference in the performance of the approximation algorithm. Due to space limitation we do not provide the corresponding plots for the BFA’s performance on the *Random* shape, which was similar to the *Convex* shape. Figure 4(b) shows that using the fastest known exact algorithms can still be efficient when the dimension is no larger than 7 or when the front consists of 20 points or fewer.

To better understand the runtime pattern of BFA in Figures 4(c) and 4(d), we drop the 30 second termination criterion and rerun the experiment for 200 points until completion. Figure 5 presents the interdependence of BFAs runtime (left plot) to the total number of Monte Carlo samplings performed by the algorithm (middle plot). We observe that for a fixed number of points, variance of the runtime increases at $d = 10$, while the average runtime slowly declines as the dimension increases (Figures 4(c) and 4(d)). Taking the exact computation usage

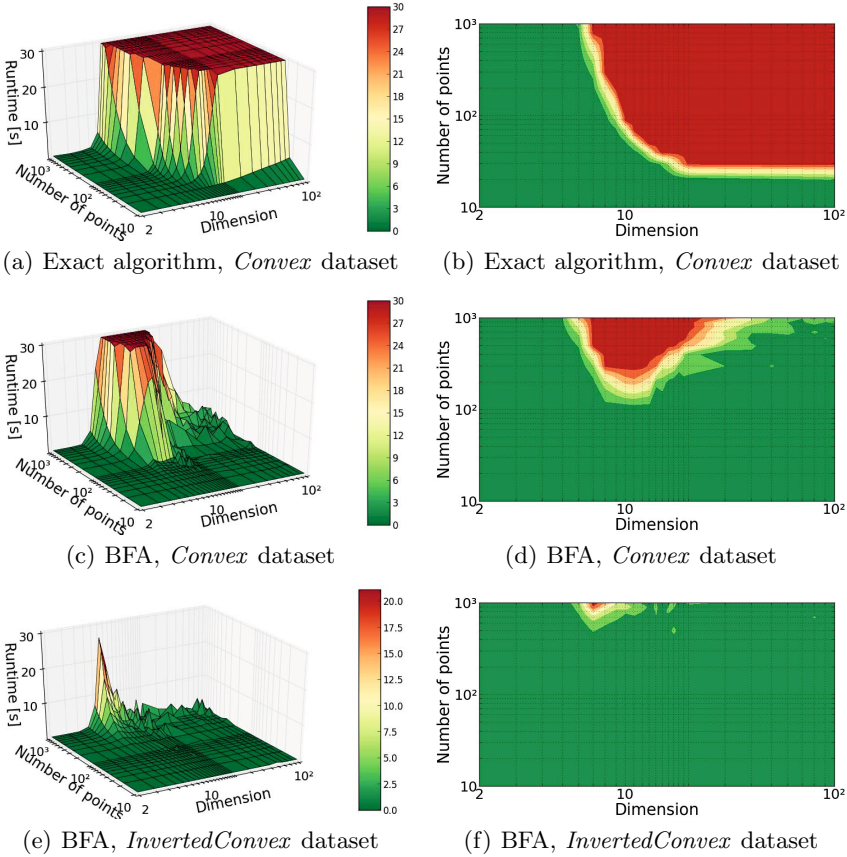


Fig. 4. Runtime of the exact algorithm – figures (a) and (b) – on *Convex* dataset, and the BFA algorithm – figures (c) to (f) – on *Convex* and *InvertedConvex* datasets

into account (rightmost plot in Figure 5) it is evident that the usage of exact computations for $d \leq 7$ amortizes the runtime in lower dimensions for BFA.

To investigate further, we pick the case with the highest runtime for each dimension and present the distribution of the number of samples over 200 points in Figure 6. We observe a dependence of the number of samples to the dimension. It is most of the time the (true) least contributor and one or two of other candidates which constitute for the majority of total number of samples, suggesting that these points remained in the race for a long time. This happens when two or more points differ very little in their contribution, which supports the impact of the *hardness of approximation*, as it was described in the original work by Bringmann and Friedrich. Surprisingly, this effect seems to be inversely proportional to the dimension (given a fixed number of points). We believe that this can be attributed to relatively sparse distribution of points as the dimension increases, leading to fewer occurrences of hard cases.

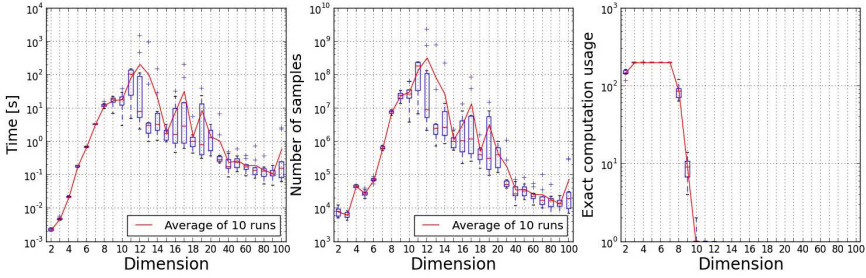


Fig. 5. Runtime of BFA algorithm (left), total number of performed sampling rounds (middle) and the number of exact computations performed by the algorithm (right) Distribution over 10 runs per dimension on *Convex* dataset of 200 points

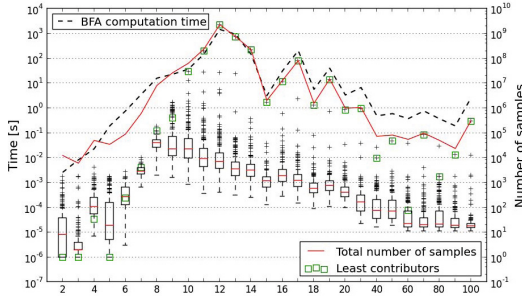


Fig. 6. Runtime of BFA algorithm (dashed line) and the distribution of the number of performed sampling rounds over 200 points (least contributors marked with a green square, red line shows the total number of samples). Maximal runtimes for each dimension picked from *Convex* datasets of 200 points.

While the high runtime has already been tied to the shape of the data, we have observed the reference point to also influence the empirical runtime performance of BFA. Figure 7 shows a runtime comparison of an exact computation using WFG and the approximation using BFA, with varying offsets applied to the nadir point. Altering the reference point influences the relative contributions of the border points, thus the least contributor can change. While the runtime of WFG seems to be indifferent to the reference point, the observed runtime of BFA spans over two orders of magnitude, in favor of smaller offsets. We suggest using a reference point relative to the size of the objective space and with a small offset (1% per objective), as we have done in our previous experiments.

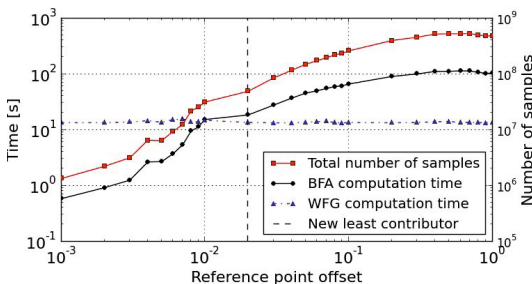


Fig. 7. Impact of the reference point offset on the runtime of WFG and BFA algorithms and the number of Monte Carlo sampling rounds. Change of the least contributor marked by a dashed line. *Concave* dataset with 100 points in 10 dimensions.

5 Conclusions

Since the currently best available exact computation algorithms quickly reach their limits for problems with 10 or more dimensions, BFA provides a superior performance as it scales much better in that regard. However, easy to overlook factors such as the geometry of the dataset or the choice of the reference point can degrade the runtime of the approximation algorithm up to two orders of magnitude, even though their observed impact on the exact methods was minimal. Although our test data was not explicitly designed to create hard cases for BFA, we observed their frequent occurrence, indicating that its runtime, while still much faster than those of exact algorithms, could be nevertheless unexpectedly high in ill-conditioned cases. By outlining the relation between the runtime of BFA and the behaviour of the underlying Monte Carlo sampling scheme, we highlight easy to overlook factors that need to be considered before employing the algorithm in practice. Taking these points into account, hypervolume approximation has a great potential for opening up previously intractable problem domains for optimization and research.

Acknowledgement. We thank Tobias Friedrich for his encouragement and helpful comments at the early stages of our research.

References

1. Fleischer, M.: The measure of Pareto optima applications to multi-objective metaheuristics. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 519–533. Springer, Heidelberg (2003)
2. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: Methods and applications, vol. 63. Shaker Ithaca (1999)
3. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7(2), 117–132 (2003)
4. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
5. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
7. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm (2001)
8. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)

9. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181(3), 1653–1669 (2007)
10. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation* 19(1), 45–76 (2011)
11. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation* 15(1), 1–28 (2007)
12. Bringmann, K., Friedrich, T.: Approximating the least hypervolume contributor: Np-hard in general, but fast in practice. *Theoretical Computer Science* 425, 104–116 (2012)
13. Bringmann, K., Friedrich, T.: Parameterized average-case complexity of the hypervolume indicator. In: *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO 2013*, pp. 575–582. ACM, New York (2013)
14. Beume, N., Fonseca, C.M., López-Ibáñez, M., Paquete, L., Vahrenhold, J.: On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation* 13(5), 1075–1082 (2009)
15. Guerreiro, A.P., Fonseca, C.M., Emmerich, M.T.: A fast dimension-sweep algorithm for the hypervolume indicator in four dimensions. In: *CCCG*, pp. 77–82 (2012)
16. Emmerich, M.T.M., Fonseca, C.M.: Computing hypervolume contributions in low dimensions: asymptotically optimal algorithm and complexity results. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) *EMO 2011*. LNCS, vol. 6576, pp. 121–135. Springer, Heidelberg (2011)
17. Bringmann, K.: An improved algorithm for Klee’s measure problem on fat boxes. *Computational Geometry* 45(5), 225–233 (2012)
18. Yildiz, H., Suri, S.: On Klee’s measure problem for grounded boxes. In: *Proceedings of the 2012 Symposium on Computational Geometry*, pp. 111–120. ACM (2012)
19. Beume, N.: S-metric calculation by considering dominated hypervolume as Klee’s measure problem. *Evolutionary Computation* 17(4), 477–492 (2009)
20. Fonseca, C.M., Paquete, L., López-Ibáñez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: *IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 1157–1163. IEEE (2006)
21. While, L., Bradstreet, L., Barone, L.: A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation* 16(1), 86–95 (2012)
22. Priester, C., Narukawa, K., Rodemann, T.: A comparison of different algorithms for the calculation of dominated hypervolumes. In: *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO 2013*, pp. 655–662. ACM, New York (2013)
23. Bringmann, K., Friedrich, T.: Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry* 43(6), 601–610 (2010)
24. Ishibuchi, H., Tsukamoto, N., Sakane, Y., Nojima, Y.: Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 527–534. ACM (2010)
25. Bringmann, K., Friedrich, T., Igel, C., Voß, T.: Speeding up many-objective optimization by Monte Carlo approximations. *Artificial Intelligence* 204, 22–29 (2013)
26. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* 10(1), 50–66 (2006)

A Portfolio Optimization Approach to Selection in Multiobjective Evolutionary Algorithms

Iryna Yevseyeva¹, Andreia P. Guerreiro²,
Michael T. M. Emmerich³, and Carlos M. Fonseca²

¹ Centre for Cybercrime and Computer Security
School of Computing Science, Newcastle University
Newcastle upon Tyne, NE1 7RU, UK
iryna.yevseyeva@ncl.ac.uk

² CISUC, Department of Informatics Engineering, University of Coimbra,
Pólo II, Pinhal de Marrocos, 3030-290 Coimbra, Portugal
{apg,cmfonsec}@dei.uc.pt

³ Leiden Institute of Advanced Computer Science, Leiden University,
Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands
emmerich@liacs.nl

Abstract. In this work, a new approach to selection in multiobjective evolutionary algorithms (MOEAs) is proposed. It is based on the portfolio selection problem, which is well known in financial management. The idea of optimizing a portfolio of investments according to both expected return and risk is transferred to evolutionary selection, and fitness assignment is reinterpreted as the allocation of capital to the individuals in the population, while taking into account both individual quality and population diversity. The resulting selection procedure, which unifies parental and environmental selection, is instantiated by defining a suitable notion of (random) return for multiobjective optimization. Preliminary experiments on multiobjective multidimensional knapsack problem instances show that such a procedure is able to preserve diversity while promoting convergence towards the Pareto-optimal front.

Keywords: Fitness assignment, portfolio selection, Sharpe ratio, evolutionary algorithms, multiobjective knapsack problem.

1 Introduction

In evolutionary algorithms (EAs), selection shapes the direction in which the search is performed by dictating which individuals are allowed to reproduce. Typically, better individuals are assigned higher fitness, and are, therefore, selected for breeding. Carrying out selection based exclusively on individual performance (e.g., proportionally to a global objective value or individual rank) may work well when a single best solution is sought, but it typically leads to an undesired loss of population diversity when searching for multiple optimal solutions to multimodal problems. For this reason, techniques such as crowding and fitness sharing [17] are used in several multiobjective EAs (MOEAs) [6,7],

to promote a good coverage of the Pareto-optimal front. Individuals located in more crowded regions are penalized, and greater chance of reproduction is given to individuals in less crowded regions.

Another approach is based on the use of quality indicators [20]. It consists of defining a notion of population quality and then inferring how much each individual contributes to the quality of the population. Algorithms such as LAHC [12], SMS-EMOA [4] and HyPE [1] instantiate this idea by using the hypervolume indicator as the measure of population quality, and clearly define what the contribution of each individual to that value is, albeit in different ways. In this work, the opposite view is adopted. An interpretation of fitness assignment as a (financial) portfolio selection problem (PSP) is proposed, where individuals are seen as assets with given (randomly distributed) monetary return values, and the fitness assigned to each individual represents an investment in that individual. In this case, it is the quality of the population that is inferred from the quality of the individuals that compose it, represented by the corresponding return distributions.

It is known from portfolio selection theory that investing only in one asset, or in similar assets, carries a risk associated with the variability of the individual returns, which is directly reflected in the variability of the overall return. Similarly, it is well known that selecting only a few of the best individuals in an EA population may lead to loss of population diversity and even to premature convergence. Therefore, the proposed analogy is completed by associating lack of population diversity with risk in the financial sense.

This paper is organized as follows. The classical PSP formulation is reviewed in the next section, leading to the proposed interpretation of fitness assignment as a portfolio selection problem. In Section 3, a new fitness assignment strategy for MOEAs is developed based on the classical PSP formulation, by specifying suitable notions of expected return and risk. This strategy is then extended to encompass solution archiving as well, allowing parental and environmental selection to be unified into a single selection problem. Preliminary experimental results on multiobjective multidimensional knapsack problem instances are presented in Section 4. The paper concludes with a discussion of the proposed approach.

2 Background

2.1 Portfolio Selection

In the classical Markowitz formulation [16] of the portfolio selection problem, asset returns are modeled as random variables, the expected values of which can usually be estimated from historical data. Risk is assessed as the variance of the overall portfolio return, and depends not only on how much individual asset returns vary, but also on how they vary in relation to one another. Thus, the covariance matrix of the joint asset return distribution is considered in addition to the expected values. A financial portfolio should optimize two conflicting objectives: maximizing the expected portfolio return and minimizing portfolio return variance. Formally:

$$\text{maximize } \sum_{i=1}^n r_i x_i = r^T x \tag{1}$$

$$\text{minimize } \sum_{i=1}^n \sum_{j=1}^n q_{ij} x_i x_j = x^T Q x \tag{2}$$

$$\text{subject to } \sum_{i=1}^n x_i = 1, \quad x_i \in [0, 1], \quad i = 1, \dots, n \tag{3}$$

where n is the number of assets, r_i is the expected return of asset i , and q_{ij} is the covariance of the returns of assets i and j . The unknown solution is represented by $x = (x_1, \dots, x_n)^T$, where each x_i denotes the proportion of capital to be invested in asset i .

Sharpe Ratio. The solution to the PSP defined by expressions (1–3) is a set of Pareto-optimal portfolios. The portfolios in this set are efficient with respect to expected return and return variance, and reflect different investor behavior: portfolios composed mostly of high-return assets are usually riskier, but simply avoiding risk is seldom profitable. Risk is usually reduced by combining assets with negatively correlated returns, although the expected return of the portfolio will necessarily decrease due to the inclusion of lower-return assets.

Several notions of an optimal return-to-risk trade-off have been proposed in the literature. Among them, the most widely used risk-adjusted performance index is the Sharpe ratio [5], also called reward-to-volatility ratio. The Sharpe ratio assesses how well the expected return of a given portfolio compensates the risk taken by measuring the excess return per unit of deviation from the mean with respect to a baseline, risk-free investment. The portfolio with the maximum Sharpe ratio, or *optimal risky portfolio*, x^* , is the solution of the following non-linear programming problem:

$$\text{maximize } \frac{r^T x - r_f}{\sqrt{x^T Q x}} \tag{4}$$

$$\text{subject to } \sum_{i=1}^n x_i = 1, \quad x_i \in [0, 1], \quad i = 1, \dots, n \tag{5}$$

where r_f is the (deterministic) return of a reference, *riskless asset*. Naturally, the expected return of an efficient portfolio should be at least r_f .

This non-convex problem can be transformed into the, easier to solve, convex quadratic programming problem:

$$\text{minimize } y^T Q y \tag{6}$$

$$\text{subject to } \sum_{i=1}^n (r_i - r_f) y_i = 1, \quad y_i \geq 0, \quad i = 1, \dots, n \tag{7}$$

by homogenizing the objective function (4), as detailed in [5]. A standard quadratic programming solver may then be used to determine the optimal risky portfolio, $x^* = y^*/k$, where $k = \sum_{i=1}^n y_i^*$.

2.2 Fitness Assignment as a Portfolio Selection Problem

In order to express fitness assignment in EAs as a portfolio selection problem, suitable analogues of individual expected return, r , and return covariance, Q , must be considered. Having established the values of these parameters, the desired fitness assignment, x , can be obtained by solving the resulting PSP, e.g. for the maximum Sharpe ratio portfolio.

In particular, when the covariance matrix Q is a scalar matrix and the expected return of each individual is set to the corresponding (single) objective value to be maximized, maximizing the Sharpe ratio will assign fitness proportionally to the difference between the individual objective values and the reference return value considered, if this difference is positive, and zero otherwise.

PSPs corresponding to other traditional fitness assignment strategies, such as linear ranking [3] and sigma-scaling [10], can also be formulated by considering ranks instead of objective values and/or appropriately selecting the value of the reference return. Consequently, the portfolio selection interpretation of fitness assignment *extends* conventional proportional fitness assignment by making the notion of *risk* explicit in the form of a covariance matrix.

The question remains of how to design the covariance matrix in order to control the loss of population diversity due to selection, while maintaining an appropriate level of selective pressure towards better solutions. One possible answer in the context of multiobjective optimization is proposed next.

3 A New Approach to Multiobjective Selection

Most current MOEAs attempt to drive the individuals in the population towards, and to distribute them across, the Pareto front of the problem, so that the final solution may be selected by a Decision Maker (DM) in an *a posteriori* fashion. In this way, modeling the (subjective) preferences of the DM is avoided, but the whole Pareto front must be approximated as well as possible in order to maximize the chance that at least one of the solutions found satisfies the DM.

3.1 Fitness Assignment

In such an *a posteriori* setting, the *return* of a given candidate solution may be seen as a random variable modeling the uncertainty associated with the unknown preferences of the DM. A simple scenario will be considered:

- Without loss of generality, the problem to be solved consists of the minimization of a d -dimensional objective function, f .
- There are $n > 1$ individuals in the population. To each individual i , $i = 1, \dots, n$, corresponds an objective vector $f_i \in \mathbb{R}^d$.
- Preferences are expressed by the Decision Maker in terms of a single goal vector, drawn from some probability distribution over the objective space. In particular, a uniform distribution on a given orthogonal range $[l, u]$ of the objective space will be assumed, where $l, u \in \mathbb{R}^d$.

- Individuals are either “acceptable” or “not acceptable” depending on whether or not they weakly dominate such a random goal vector, respectively. Therefore, each individual will be deemed acceptable with a certain probability, and the corresponding return (or *acceptability*) is a Bernoulli random variable taking a value of 1 if the solution is acceptable and 0 otherwise.

Under these conditions, the expected return r_i of an individual i is equal to the proportion of the (given orthogonal range of the) objective space which f_i dominates, i.e.,

$$r_i = p_i = \frac{\lambda([f_i, \infty[\cap [l, u])}{\lambda([l, u])} \tag{8}$$

where $\lambda(\cdot)$ denotes the Lebesgue measure (or hypervolume) of the given region, and $[f_i, \infty[$ is the region dominated by f_i . Since returns are Bernoulli distributed, the return covariance for a pair of individuals is

$$q_{ij} = p_{ij} - p_i p_j \tag{9}$$

$$= \frac{\lambda([f_i, \infty[\cap [f_j, \infty[\cap [l, u])}{\lambda([l, u])} - p_i p_j \tag{10}$$

$$= \frac{\lambda([(f_i \vee f_j), \infty[\cap [l, u])}{\lambda([l, u])} - p_i p_j \tag{11}$$

where $f_i \vee f_j$ denotes the *join*, or componentwise maximum, of objective vectors f_i and f_j , for $i, j = 1, \dots, n$. Note that $p_{ii} = p_i$, and that $q_{ii} = p_i - p_i^2$ is simply the variance of the return of individual i . As a consequence, the return of a riskless asset must be zero ($r_f = 0$) under this model.

The above expressions show that the return covariance relates the size of the region simultaneously attained by two individuals to the sizes of the regions attained individually by each one, an idea which is also at the heart of the definition of an extended dominance relation known as volume dominance [13], although the details of the two methods are considerably different. Whereas the aim of volume dominance is to establish whether an individual should be considered better than another, here the aim is to gauge the (dis)similarity between individuals. Positive covariance indicates that two individuals attain much the same region of the objective space, whereas negative covariance is a sign that the regions attained do not overlap much. Since portfolio variance is reduced by combining negatively correlated assets, and greater returns are preferred, a risk-adjusted portfolio should consist of a diverse set of individuals along the non-dominated front.

Illustrative examples are presented in Fig. 1. When all individuals are non-dominated and evenly distributed on a linear front (left), the fitness assigned to each one by maximizing the Sharpe ratio is the same. On the other hand, if they are not evenly distributed (center), isolated individuals are assigned greater fitness values than those in crowded regions of the front. In the more general case (right), dominated individuals, as well as some non-dominated individuals, are assigned zero fitness, whereas the remaining non-dominated individuals are

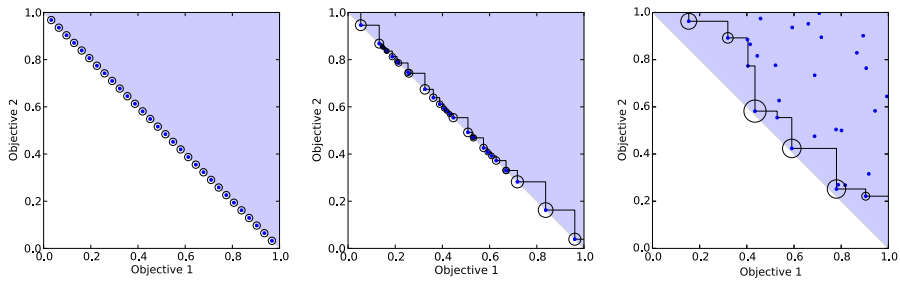


Fig. 1. Maximum Sharpe ratio fitness assignment. Left and center: 30 non-dominated individuals. Right: an arbitrary 30-individual population. Circle area is proportional to assigned fitness. $l = (0, 0)$, $u = (1, 1)$.

assigned high fitness values. Therefore, the optimal risky portfolio may not maximize the hypervolume indicator on an arbitrary front, although it would seem to correctly identify at least some interesting non-dominated solutions.

3.2 Environmental Selection and Archiving

The portfolio selection model described so far is aimed at *parental* selection, i.e., at the selection of individuals for breeding. Once fitness has been assigned, this is typically achieved with a sampling procedure such as roulette-wheel selection (RWS) [8] or stochastic universal sampling (SUS) [2].

Environmental selection, on the other hand, is aimed at replacing old individuals with new ones, and may be performed based on fitness, the number of iterations an individual has survived in the population, or even randomly. Also, offspring may replace the parents unconditionally or depending on whether they outperform them. A replacement strategy where the best individuals always survive is known as elitist.

In contrast to the single objective case, the implementation of elitist environmental selection in the multiobjective case must deal with the possible incomparability between individuals. If only dominated individuals are ever replaced by new ones, either the population is allowed to grow indefinitely or the algorithm will terminate as soon as all individuals in the population are non-dominated. For this reason, alternative environmental selection strategies where non-dominated individuals may be replaced by new ones in order to keep the size of the population constant have been proposed and extensively studied [12,11,15]. Because the population acts as an archive of non-dominated solutions, such strategies have become known as *bounded archiving* strategies.

MOEAs typically implement environmental selection separately from parental selection. However, the two can be meaningfully combined into a single portfolio selection problem with a cardinality constraint. Assuming a parental population size of n and the production of m offspring at each iteration, the new problem consists of assigning non-zero fitness to at most n individuals from the $n + m$

parents and offspring currently available. In this way, environmental selection is performed so as to maximize the Sharpe ratio of the resulting portfolio. It is not difficult to see that this approach guarantees that the Sharpe ratio may never decrease from one iteration to the next. Therefore, this combined environmental selection and fitness assignment mechanism implements a monotonic bounded archiving strategy [15].

Unfortunately, portfolio selection with cardinality constraints is no longer a convex optimization problem, and may be difficult to solve exactly (it is generally NP-hard). In practice, however, this will depend on how tightly constrained a particular instance turns out to be. Indeed, solving the relaxed problem will, in many cases, lead to a solution that satisfies the cardinality constraint, unless the population is so well distributed that more than n individuals would, in principle, be assigned non-zero fitness. This simply requires that m is chosen to be sufficiently small in comparison to n .

4 Experimental Results

The proposed approach to multiobjective selection was implemented in an otherwise conventional mutation-selection evolutionary algorithm with population size $n = 200$. The parental population was sampled proportionally to assigned fitness using SUS [2] to select $m = 50$ parents that were mutated to produce m offspring. For simplicity, no recombination operator was used.

The algorithm was applied to multiobjective knapsack instances from [19] with 100 and 500 items, with 2, 3 and 4 objectives, and as many capacity constraints as objectives. For the purpose of constructing the portfolio selection problems, knapsack values and weights were taken to range from zero to their maximum values (when all items are included). The return distribution of each individual was computed as described in section 3.1, but using the preferability relation [7] instead of dominance, in order to accommodate constraints and objectives in the same formulation. Additional (linear) constraints on fitness were imposed so that no individual could expect to reproduce more than once in each generation.

Individuals were represented as binary strings, and mutation consisted of either flipping one bit at random or exchanging two randomly-selected bits of different value, so as to uniformly sample the resulting 1-flip-exchange neighborhood [14]. In order to study the long-run behavior of the algorithm, here referred to as Portfolio Optimization Selection Evolutionary Algorithm (POSEA), and account for its stochasticity, 13 long runs with 1 million function evaluations each were performed for each instance. For comparison purposes, equally long runs were performed using SPEA2 [18], NSGA-II [6] and SMS-EMOA [4] with the same population size ($n = 200$) and mutation operator. As before, no recombination was used. The number of offspring per generation, m , was set to the default in each case: $m = n$ in SPEA2 and NSGA-II and $m = 1$ in SMS-EMOA.

Experimental results are presented in Figs. 2 and 3, where the maximum, median and minimum of 13 runs are shown for each algorithm on 2-, 3- and 4-objective instances. SPEA2 and NSGA-II do not use hypervolume information, and tend to perform worse or, at most, slightly better than SMS-EMOA

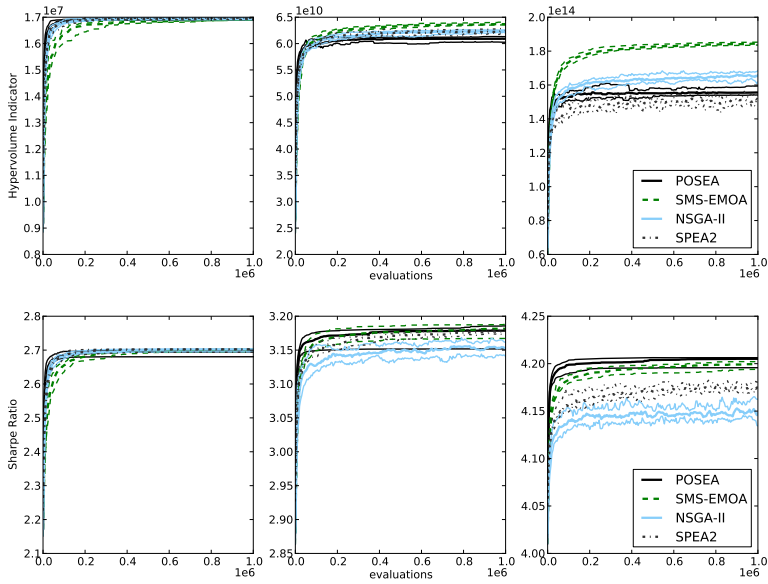


Fig. 2. Performance on 100-item knapsack instances: 2 objectives (left), 3 objectives (center) and 4 objectives (right)

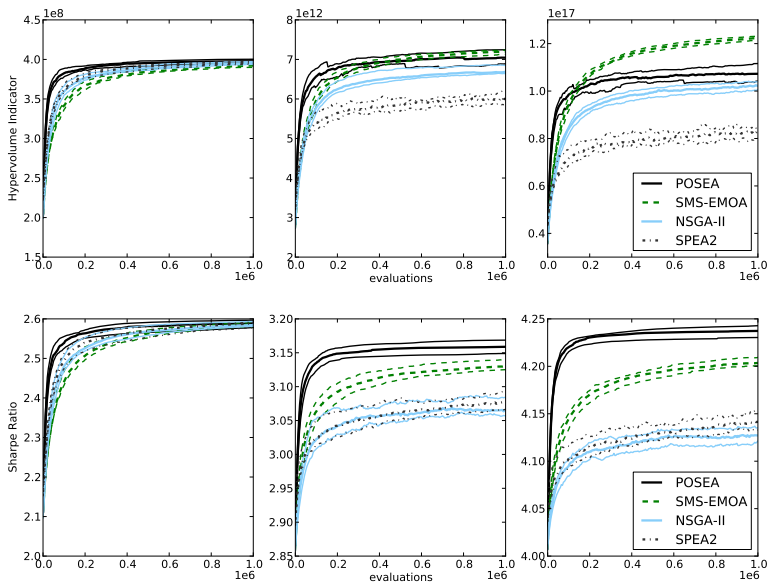


Fig. 3. Performance on 500-item knapsack instances: 2 objectives (left), 3 objectives (center) and 4 objectives (right)

and POSEA with respect to both the hypervolume indicator and the Sharpe ratio. SMS-EMOA can be seen to perform best with respect to the hypervolume indicator, whereas POSEA clearly does not cover the Pareto-front as well as SMS-EMOA. On the other hand, SMS-EMOA achieves lower or, at most, similar values of Sharpe ratio, indicating that POSEA focused on the most relevant non-dominated solutions according to the DM model adopted.

5 Conclusions

In this work, a fitness assignment approach based on (financial) portfolio optimization was proposed. By modeling the uncertainty associated with Decision Maker preferences probabilistically, the quality (or return) of each individual solution becomes a random variable, and fitness assignment consists of forming a portfolio of individuals balancing overall expected return against return variance, e.g., based on the Sharpe ratio.

Although the probabilistic Decision Maker model adopted in this work is rather simplistic, empirical evidence suggests that it possesses some interesting properties, such as not favoring dominated solutions over non-dominated ones and promoting diversity in the population, even if it does not maximize the hypervolume indicator in the general case. A theoretical study of these and other properties is currently under way [9], but more experimentation is required to evaluate the performance of POSEA and how the number of offspring, m , may influence it. Furthermore, since the size of the quadratic programming problem to be solved at each generation is independent of the number of objectives, the method is potentially much faster than hypervolume-based selection (depending on m), especially as the number of objectives grows.

The proposed approach establishes a bridge between multiobjective selection and optimization under uncertainty. By considering alternative probabilistic DM models based on other indicators and/or preference articulation strategies, the portfolio optimization paradigm should contribute to unifying solution-oriented preferences and set-oriented preferences under a common framework.

Acknowledgments. I. Yevseyeva's research was funded by the Academy of Finland (grant 126476) and the EC Erasmus Mundus ECW Lot 6 project at the Faculty of Science and Technology of the University of Algarve, Portugal. A. P. Guerreiro acknowledges Fundação para a Ciência e a Tecnologia (FCT) for Ph.D. studentship SFHR/BD/77725/2011, co-funded by the European Social Fund and by the State Budget of the Portuguese Ministry of Education and Science in the scope of NSRF-HPOP-Type 4.1-Advanced Training.

References

1. Bader, J., Zitzler, E.: HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation* 19(1), 45–76 (2011)
2. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: *Proc. Second International Conference on Genetic Algorithms*, pp. 14–21 (1987)

3. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proc. First International Conference on Genetic Algorithms, pp. 101–111 (1985)
4. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. *EJOR* 181, 1653–1669 (2007)
5. Cornuejols, G., Tutuncu, R.: *Optimization Methods in Finance*. Cambridge University Press (2007)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
7. Fonseca, C.M., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part I: A unified formulation. *IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans* 28(1), 26–37 (1998)
8. Goldberg, D.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading (1989)
9. Guerreiro, A.P.: *Portfolio Selection in Evolutionary Algorithms*. Ph.D. thesis proposal, University of Coimbra, Coimbra, Portugal (2012)
10. Hancock, P.J.B.: An empirical comparison of selection methods in evolutionary algorithms. In: Fogarty, T.C. (ed.) *AISB-WS 1994*. LNCS, vol. 865, pp. 80–94. Springer, Heidelberg (1994)
11. Knowles, J., Corne, D.: Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation* 7(2), 100–116 (2003)
12. Knowles, J., Corne, D., Fleisher, M.: Bounded archiving using the Lebesgue measure. In: Proc. IEEE Congress on Evolutionary Computation (CEC 2003), vol. 4, pp. 2490–2497. IEEE Press, New York (2003)
13. Le, K., Landa-Silva, D.: Obtaining better non-dominated sets using volume dominance. In: *IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 3119–3126 (September 2007)
14. Liefvooghe, A., Paquete, L., Figueira, J.R.: On local search for bi-objective knapsack problems. *Evolutionary Computation* 21(1), 179–196 (2013)
15. López-Ibáñez, M., Knowles, J., Laumanns, M.: On sequential online archiving of objective vectors. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) *EMO 2011*. LNCS, vol. 6576, pp. 46–60. Springer, Heidelberg (2011)
16. Markowitz, H.: Portfolio selection. *Journal of Finance* 7(1), 77–91 (1952)
17. Sareni, B., Krahenbuhl, L.: Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation* 2(3), 97–106 (1998)
18. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. In: *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN 2001)*, pp. 95–100 (2002)
19. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms — A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) *PPSN 1998*. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
20. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7(2), 117–132 (2003)

Using a Family of Curves to Approximate the Pareto Front of a Multi-Objective Optimization Problem

Saúl Zapotecas Martínez¹, Víctor A. Sosa Hernández², Hernán Aguirre¹,
Kiyoshi Tanaka¹ and Carlos A. Coello Coello²

¹ Faculty of Engineering, Shinshu University, 4-17-1 Wakasato, Nagano, 380-8553, Japan
saul.zapotecas@gmail.com, {ahernan, ktanaka}@shinshu-u.ac.jp

² Computer Science Department, CINVESTAV-IPN, Av. IPN 2508, C.P. 07360,
San Pedro Zacatenco, Mexico D.F., Mexico
msosa@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

Abstract. The design of selection mechanisms based on quality assessment indicators has become one of the main research topics in the development of Multi-Objective Evolutionary Algorithms (MOEAs). Currently, most indicator-based MOEAs have employed the hypervolume indicator as their selection mechanism in the search process. However, hypervolume-based MOEAs become inefficient (and eventually, unaffordable) as the number of objectives increases. In this paper, we study the construction of a reference set from a family of curves. Such reference set is used together with a performance indicator (namely Δ_p) to assess the quality of solutions in the evolutionary process of an MOEA. We show that our proposed approach is able to deal (in an efficient way) with problems having many objectives (up to ten objective functions). Our preliminary results indicate that our proposed approach is highly competitive with respect to two state-of-the-art MOEAs over the set of test problems that were adopted in our comparative study.

1 Introduction

In spite of the success of Multi-Objective Evolutionary Algorithms (MOEAs) for solving engineering and scientific problems, their application in problems with many objectives continues to be a hot research topic. In the last decade, several indicator-based MOEAs have been proposed [1,13,16]. The main motivation for using indicator-based MOEAs is that Pareto optimality quickly degrades its performance as we increase the number of objectives. One of the most popular indicator-based MOEAs of today is the *S Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA)* [1], which is based on the use of the hypervolume. However, SMS-EMOA has an important disadvantage: computing the hypervolume in high dimensionality (i.e., for problems with 4 or more objectives) is computationally expensive and quickly becomes unaffordable.

This has led to the use of other quality indicators such as: $R2$ [10] and Δ_p [14]. For these two indicators, it is possible to use a reference set in order to compute such metrics.¹ In fact, it is absolutely necessary the definition of a reference set for the Δ_p indicator. In the specialized literature, most authors working with indicator-based MOEAs,

¹ The version of $R2$ using a reference set is called $R2_R$ in [10].

have preferred the use of $R2$ (see e.g., [2,7,11]), while the use of Δ_p has been scarcely explored (see e.g. [8,13]). This is, perhaps, because it is easier to formulate a set of cost functions (another form of using $R2$) than to define a reference set (this requires an appropriate discretization of the real Pareto front). Since the features of the real Pareto front of a multi-objective optimization problem (MOP) are unknown, the construction of an appropriate reference set becomes a real challenge for the design of MOEAs based on reference sets. Some authors have defined the reference set by constructing segments of the possible Pareto front employing information of the nondominated solutions found along the search process, see e.g. [13]. Nonetheless, the construction of a generalized structure (a generalized reference set) constitutes a field still unexplored. In this paper, we propose the *Reference Indicator-Based Evolutionary Multi-Objective Algorithm (RIB-EMOA)*, which is based on Δ_p [14] and builds a reference set by using a family of curves. Our proposed approach is compared with respect to two other MOEAs using standard test problems having between three and ten objectives.

The remainder of this paper is organized as follows. In Section 2, we present the basic concepts required to understand this paper. In Section 3, we explain the general framework of our proposed approach. The detailed description of the construction of the reference set is presented in Section 4. In Section 5, we present the validation of our proposed approach. Finally, the conclusions and some possible paths for future research are drawn in Section 6.

2 Basic Concepts

2.1 Multi-objective Optimization

Assuming minimization, a continuous MOP can be formulated as:

$$\min_{\mathbf{x} \in \Omega} F(\mathbf{x}) \quad (1)$$

where $\Omega \subset \mathbb{R}^n$ defines the decision space and $F : \Omega \rightarrow \mathbb{R}^k$ is defined as the vector of continuous functions, such that each $f_j : \Omega \rightarrow \mathbb{R}$ ($j = 1, \dots, k$) represents the function to be minimized. In this paper, we consider the box-constrained case, i.e., $\Omega = \prod_{i=1}^n [a_i, b_i]$. Therefore, each variable vector $\mathbf{x} = (x_1, \dots, x_n)^T \in \Omega$ is such that $a_i \leq x_i \leq b_i$ for all $i \in \{1, \dots, n\}$.

- Definition 1.** a) Let $\mathbf{x}, \mathbf{y} \in \Omega$. Then the vector \mathbf{x} “dominates” \mathbf{y} (denoted by $\mathbf{x} < \mathbf{y}$) with respect to the problem (1), if and only if: i) $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$ for all $i \in \{1, \dots, k\}$; and ii) $f_j(\mathbf{x}) < f_j(\mathbf{y})$ for at least one $j \in \{1, \dots, k\}$.
- b) Let $\mathbf{x}^* \in \Omega$, we say that \mathbf{x}^* is a “Pareto optimal” solution, if there is no other solution $\mathbf{y} \in \Omega$ such that $\mathbf{y} < \mathbf{x}^*$.
- c) The Pareto set (PS) of problem (1) is defined by: $PS = \{\mathbf{x} \in \Omega : \mathbf{x} \text{ is a Pareto optimal solution}\}$ and the Pareto front (PF) is defined by: $PF = \{F(\mathbf{x}) : \mathbf{x} \in PS\}$.

2.2 Δ_p Indicator

The Δ_p indicator [14], can be viewed as the Hausdorff distance between an approximation set and the real PF of a MOP. This indicator is defined by a slight modification


```

Input:
a stopping criterion;
N: the population size;
Output:
 $P_t$ : the final approximation to the PF.


---


1  $t = 0$ ;
  /* Initialize a population of N individuals */
2  $P_t = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ ;
3 while stopping criterion is not satisfied do
  /* Generate a trial solution */
4    $\mathbf{q} = \text{generate}(P_t)$ ;
  /* Select the N best individuals */
5    $P_{t+1} = \text{reduce}(P_t \cup \{\mathbf{q}\})$ ;
6    $t = t + 1$ ;
7 end

```

Algorithm 1: RIB-EMOA

```

Input:
Q: the population to be reduced;
Output:
 $Q^*$ : the reduced population.


---


1  $P^* = \text{nondominated\_solutions}(Q)$ ;
2 if  $P^* \neq Q$  then
3    $s = \arg \max_{y \in Q} d(y, Q)$ ;
4 else
5    $s = \arg \max_{y \in Q} \Psi(y, Q, R)$ ;
6 end
7  $Q^* = Q \setminus \{s\}$ ;
8 return ( $Q^*$ )

```

Algorithm 2: $\text{reduce}(Q)$

from the well-known quality indicators: Generational Distance (GD) [15] and Inverted Generational Distance (IGD) [3]. Formally, the Δ_p indicator can be defined as follows.

Definition 2. Let $P = \{\mathbf{x}^1, \dots, \mathbf{x}^{|P|}\}$ an approximation and $R = \{\mathbf{r}^1, \dots, \mathbf{r}^{|R|}\}$ be a discretization of the real PF of a MOP. The “ Δ_p indicator” is defined as:

$$\Delta_p(P, R) = \max\{I_{GD_p}(P, R), I_{IGD_p}(P, R)\} \quad (2)$$

where I_{GD_p} and I_{IGD_p} are a slight modification from GD and IGD, respectively. They are defined as: $I_{GD_p}(P, R) = \left(\frac{1}{|P|} \sum_{i=1}^{|P|} d_i^p\right)^{\frac{1}{p}}$ and $I_{IGD_p}(P, R) = \left(\frac{1}{|R|} \sum_{j=1}^{|R|} \hat{d}_j^p\right)^{\frac{1}{p}}$, where d_i and \hat{d}_j are: the Euclidean distance from \mathbf{x}^i to its closest member $\mathbf{r} \in R$, and the Euclidean distance from \mathbf{r}^j to its closest member $\mathbf{x} \in P$, respectively.

Therefore, a low Δ_p value means that the set P has a better approximation to the real PF. More details of the Δ_p indicator and its properties can be found in [14].

3 The Reference Indicator-Based EMOA

3.1 General Framework

RIB-EMOA initializes a population P_t ($t = 0$) of N randomly generated individuals. A new individual \mathbf{q} is generated by choosing (in a random way) two different parents from P . The parents are recombined by means of Simulated Binary Crossover (SBX) and the resulting children are mutated using Polynomial-Based Mutation (PBM) [5].² The new individual \mathbf{q} (defined by any child) becomes a member of the next population P_{t+1} , if replacing another individual leads to a higher quality of the population in terms of the Δ_p indicator. The general framework of RIB-EMOA is presented in Algorithm 1. In the following sections, we will explain the reduction procedure (in Algorithm 1) and the proposed reference set construction procedure.

² However, the use of any other evolutionary operators is also possible.

3.2 Reduction Procedure

The procedure “reduce” (in Algorithm 1) selects the N best individuals from $Q = P_t \cup \{\mathbf{q}\}$ using the Δ_p indicator and a discretization of the real PF (denoted as R). In the following description, let us consider P^* and $d(\mathbf{y}, Q)$ as the set of nondominated solutions in Q and the number of points from Q that dominate solution $\mathbf{y} \in Q$, respectively. More formally, $d(\mathbf{y}, Q) = |\{\mathbf{x} \in Q : \mathbf{x} < \mathbf{y}, \mathbf{x} \neq \mathbf{y}\}|$. Since the cardinality of Q is $N+1$, one solution from Q needs to be discarded. The following definition is introduced.

Definition 3. Let R be a discretization of the real PF of a MOP. The exclusive contribution of a solution $\mathbf{y} \in Q$ to the Δ_p indicator is defined as:

$$\Psi(\mathbf{y}, Q, R) = \Delta_p(Q \setminus \{\mathbf{y}\}, R) \tag{3}$$

Clearly, if $P^* = Q$, then all solutions in Q are nondominated and all of them are equally efficient in terms of Pareto optimality. In this case, we discard the solution $\mathbf{s} \in Q$ such that it maximizes the contribution to the Δ_p indicator, that is: $\mathbf{s} = \arg \max_{\mathbf{y} \in Q} \Psi(\mathbf{y}, Q, R)$. On the other hand, if $P^* \neq Q$ then there exist solutions in Q dominated by any solution in P^* . In this case, we discard the solution with the highest $d(\mathbf{y}, Q)$ value instead of using the Δ_p indicator. With that, the computation of Δ_p is avoided, thus reducing the computational cost of RIB-EMOA. Algorithm 2 shows the general reduce procedure.

Since we do not have any information related to the real PF of the MOP, the discretization of the reference set (R) is carried out by generating an artificial surface which should be a proper representation to the real PF . The next section will explain (in detail) the construction of such reference surface.

4 Reference Set Construction

4.1 Pareto Front Families

In real-world applications, there exist several problems for which the features of the real PF of a MOP are unknown. However, such PF could describe a convex or concave curve in objective space; otherwise, the PF could draw a linear surface where there is neither concavity nor convexity. Without loss of generality, we will assume that the PF of a MOP is normalized in the range $[0, 1]$, i.e., $0 \leq f_j \leq 1$, for each $j \in \{1, \dots, k\}$. Then, we associate such PF to a curve of the following family:

$$\{(y_1)^\alpha + \dots + (y_k)^\alpha = 1 : y_j \in [0, 1], \alpha \in (0, \infty)\} \tag{4}$$

This family of curves possesses the following properties: 1) if $\alpha > 1$, the curve is concave; 2) if $\alpha < 1$, the curve is convex; and 3) if $\alpha = 1$, a linear surface is defined. Clearly, if $\alpha = 1$, then each vector $\mathbf{y}^i = (y_1^i, \dots, y_k^i)^T$ in the surface satisfies $\sum_{j=1}^k y_j^i = 1$ and $y_j^i \geq 0$, for each $i \in \{1, \dots, \mu\}$. In other words, if $\alpha = 1$, the surface will describe a set of weight vectors, which could be much easier to be discretized.

Remark 1. Let $C = \{\mathbf{c}^1, \dots, \mathbf{c}^\mu\}$ be a set of μ weight vectors in \mathbb{R}^k , i.e., each $\mathbf{c}^i = (c_1^i, \dots, c_k^i)^T$ satisfies $\sum_{j=1}^k c_j^i = 1$ and $c_j^i \geq 0$, for each $i \in \{1, \dots, \mu\}$. Then, the vector:

$$\mathbf{y}^i = (c_1^i / \|\mathbf{c}^i\|_\alpha, \dots, c_k^i / \|\mathbf{c}^i\|_\alpha)^T \tag{5}$$

satisfies equation (4), where $\|\cdot\|_\alpha$ denotes the α -norm function.

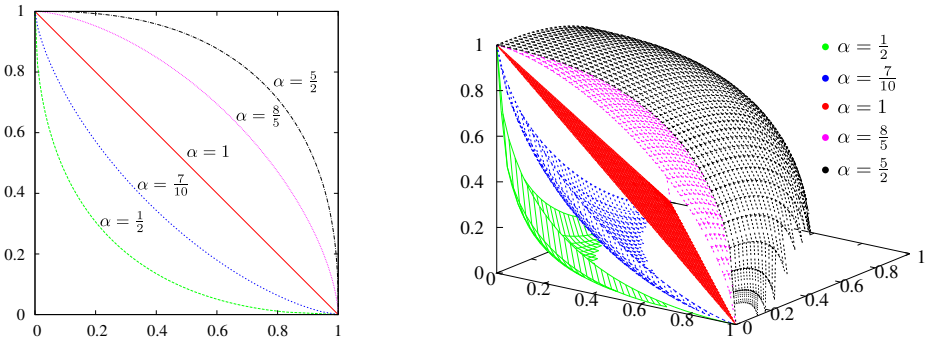


Fig. 1. Reference curves for $\alpha = \frac{1}{2}, \frac{7}{10}, 1, \frac{8}{5}, \frac{5}{2}$ in MOPs with two and three objective functions

The above remark motivates the construction of a set of weight vectors. Moreover, if the weight vectors are properly distributed in \mathbb{R}^k , a proper representation of the reference curve could be reached. Therefore, the weight set shall be the starting point for the construction of different curves by using equation (4). In Figure 1, we show different curves for different α values in problems with two and three objectives, respectively.

4.2 Weights Set

In the specialized literature, there are several strategies for generating weight vectors in an Euclidean space. Among these techniques, the Uniform Design (UD) method [9] has shown to be an effective technique in the design of weight vectors properly scattered. However, this technique becomes inefficient when the dimensionality (k) and the cardinality (μ) of the weights set increase; it has a computational complexity of $O(\binom{\mu}{k})$. Since we use the reference surface (constructed from the weights set) to compute the Δ_p indicator, for a better quality of measurement, the number of elements in the discretized reference surface should be greater than the approximation to the PF given by the MOEA. Here, we generate the weights set by using a low discrepancy sequence based on lattices, whose complexity is given by $O(\mu \times k)$, which is far lower than the one given by the UD method.

Let d be the dimension of each vector in the low discrepancy sequence. Then, the i^{th} vector in the sequence (for $i = 1, \dots, \mu$) is defined by:

$$(i/k, \{i \times \rho_1\}, \dots, \{i \times \rho_{d-1}\}) \tag{6}$$

where $\rho_1, \dots, \rho_{d-1}$ are $d-1$ distinct irrational numbers and $\{\cdot\}$ denotes the fractional part of the real value (modulo-one arithmetic). In this work, we stated each ρ_l by the transcendental number $\varphi = \frac{\sqrt{5}+1}{2}$ (the golden section), i.e., $\rho_l = \varphi$ for each $l \in \{1, \dots, d-1\}$.

Let $B^{k-1} = [0, 1]^{k-1}$ be the $(k-1)$ -dimensional design space in a unit cube. Let $B^* = \{\mathbf{b}^1, \dots, \mathbf{b}^\mu\}$ be the lattice-based low discrepancy sequence on B^{k-1} generated by equation (6). Each weight vector $\mathbf{c}^i = (c_1^i, \dots, c_k^i)^T \in C = \{\mathbf{c}^1, \dots, \mathbf{c}^\mu\}$ is achieved by employing each $\mathbf{b}^i = (b_1^i, \dots, b_{k-1}^i)^T \in B^*$ according to the following equation:

$$c_j^i = \begin{cases} \left(1 - (b_j^i)^{\frac{1}{k-j}}\right) \prod_{l=1}^{j-1} (b_l^i)^{\frac{1}{k-l}}, & \text{if } j \in \{1, \dots, k-1\} \\ \prod_{l=1}^{k-1} (b_l^i)^{\frac{1}{k-l}}, & \text{if } j = k \end{cases} \quad (7)$$

The above transformation, satisfies $\sum_{j=1}^k c_j^i = 1, c_j^i \geq 0$, for each $i \in \{1, \dots, \mu\}$ [9].

4.3 Reference Surface Construction

After obtaining the weights set (C) , the reference surface is constructed by finding the α value that will transform the set C in an appropriate curve for a determined MOP. The following definition is relevant.

Definition 4. Let \mathbf{x}_j^* be the respective global minimizers of $f_j(\mathbf{x})$, $j = 1, \dots, k$ over $\mathbf{x} \in \Omega$. Let $F_j^* = F(\mathbf{x}_j^*)$, $j = 1, \dots, k$. Let Φ be the $k \times k$ matrix whose j^{th} column is $F_j^* - F^*$. Then, the set of points in \mathbb{R}^k that are convex combinations of F_j^* , i.e., $\mathcal{H} = \{\Phi\beta : \beta \in \mathbb{R}^k, \sum_{j=1}^k \beta_j = 1, \beta_j \geq 0\}$ is referred to as the Convex Hull of Individual Minima (CHIM) [4].

In the above definition, $F^* = (f_1^*, \dots, f_k^*)^T$ denotes the utopian vector defined by the global minima values of each objective function f_j .

Let us consider $Q = P_t \cup \{\mathbf{q}\}$ as the current approximation to the real PF achieved by RIB-EMOA. Then, we state the extremes (individual minima) of the PF (denoted by ξ^i 's, for each $i \in \{1, \dots, k\}$) according to the following achievement function.

$$\xi^i = \arg \min_{\mathbf{x} \in Q} \max_{j=1}^k \left((f_j(\mathbf{x}) - f_j^*) / e_j^i \right) \quad (8)$$

where $\mathbf{e}^i = (e_1^i, \dots, e_k^i)^T$ is the canonical basis in \mathbb{R}^k (i.e., \mathbf{e}^i denotes the vector with a 1 in the i^{th} coordinate and 0 elsewhere). Each f_j^* is stated by the minimum value of the j^{th} objective function found along the search process. For $e_j^i = 0$, we use $e_j^i = 1 \times 10^{-6}$.

Let us consider $H^b = (\mathbf{z}^b, \mathbf{n}^b)$ as the hyper box formed by the extreme vectors $\{\xi^1, \dots, \xi^k\}$. More precisely, the hyper box H^b is defined by the vectors $\mathbf{z}^b = (z_1, \dots, z_k)^T$ and $\mathbf{n}^b = (n_1, \dots, n_k)^T$, such that: $z_j = \min_{i=1}^k \xi_j^i$ and $n_j = \max_{i=1}^k \xi_j^i$, for each $j \in \{1, \dots, k\}$. Then, the computation of α for the creation of the reference surface takes place according to the following description.

Let $A = \{\mathbf{x}^1, \dots, \mathbf{x}^{|A|}\}$ be the set of nondominated solutions from Q such that each solution vector $F(\mathbf{x}^i) = (f_1(\mathbf{x}^i), \dots, f_k(\mathbf{x}^i))^T$ is contained in the hyper box H^b . We consider that each solution vector $F(\mathbf{x}^i)$ is normalized in $[0, 1]$, and it will be denoted as $\hat{F}(\mathbf{x}^i) = (\hat{f}_1(\mathbf{x}^i), \dots, \hat{f}_k(\mathbf{x}^i))^T$, for each $i \in \{1, \dots, |A|\}$. Then, the convex hull \mathcal{H} in the normalized space corresponds to be a set of weight vectors and we denote to this as $\hat{\mathcal{H}}$.

The α value is stated by finding the solution vector $\hat{F}(\mathbf{x}^b)$ which describes the maximum bulge (sometimes called “knee”) formed by the convex hull $\hat{\mathcal{H}}$ and the solution vectors $\hat{F}(\mathbf{x}^i)$'s. We state this solution (\mathbf{x}^b) such that it minimizes a Tchebycheff problem. To be more precise: $\mathbf{x}^b = \arg \min_{\mathbf{x} \in A} \max_{1 \leq j \leq k} \{\lambda_j |\hat{f}_j(\mathbf{x}) - f_j^*|\}$ with the weight vector $(\lambda_1 = \frac{1}{k}, \dots, \lambda_k = \frac{1}{k})^T$, where k denotes the number of objective functions.

In order to ensure that the reference curve will touch the maximum bulge, it is initially defined by finding the α value which satisfies equation (4) for the solution vector $\hat{F}(\mathbf{x}^b)$. In other words:³ $\alpha = \arg \min_{\hat{\alpha} \in (0, \infty)} \hat{f}_1(\mathbf{x}^b)^{\hat{\alpha}} + \dots + \hat{f}_k(\mathbf{x}^b)^{\hat{\alpha}} - 1$.

Let us consider the weights set C as an appropriate discretization of $\hat{\mathcal{H}}$. Then, the construction of the reference surface $R = \{\mathbf{y}^1, \dots, \mathbf{y}^\mu\}$ is carried out by transforming each weight vector $\mathbf{c}^i \in C$ according to equation (5), for each $i \in \{1, \dots, \mu\}$. This transformation does not guarantee that all the elements in R dominate to all solution vectors $\hat{F}(\mathbf{x}^i)$, for each $\mathbf{x}^i \in A$. However, since the surface (R) intersects the maximum bulge (i.e., it passes through the point $\hat{F}(\mathbf{x}^b)$) and all solutions in A are nondominated, most solutions in A should be dominated by R . Nevertheless, the reference surface is fixed to dominate all the solutions in A .

For each solution vector $\hat{F}(\mathbf{x}^i)$ there exists a vector $\mathbf{h}^i = (h_1^i, \dots, h_k^i)^T$ with direction $\hat{F}(\mathbf{x}^i)$ (from the origin) such that \mathbf{h}^i is a weight vector, i.e., $\mathbf{h}^i \in \hat{\mathcal{H}}$. Such weight vector can be reached by $h_j^i = \hat{f}_j(\mathbf{x}^i) / \sum_{j=1}^k \hat{f}_j(\mathbf{x}^i)$, for each $i \in \{1, \dots, |A|\}$ and $j \in \{1, \dots, k\}$. Then, before computing the transformation of the whole weights set, we verify if \mathbf{h}^i under the transformation of α in equation (5) (denoted by \mathbf{h}_α^i) is dominated by $\hat{F}(\mathbf{x}^i)$. In such case (i.e., if $\hat{F}(\mathbf{x}^i) < \mathbf{h}_\alpha^i$), a new search of α needs to be conducted, but using the solution vector $\hat{F}(\mathbf{x}^i)$ instead of $F(\mathbf{x}^b)$. Finally, the normalized surface R is translated to the utopian vector F^* and scaled to the individual minima ξ 's, i.e., in the original objective space.

5 Experimental Study

In order to assess the performance of our proposed approach, we compared its results with respect to those obtained by SMS-EMOA and a version of SMS-EMOA that uses Monte Carlo simulations to approximate the S metric (we called it HyPE-EMOA). We adopted the seven unconstrained MOPs from the well-known DTLZ test suite [6]. Due to space limitations and the known geometrical shapes of each DTLZ problem, we compare herein, the performance of each algorithm by using only the Generational Distance (GD) [15]. The GD for DTLZ1 was computed as $GD = \frac{1}{|P|} \sum_{\mathbf{x} \in P} \|F(\mathbf{x})\|_1 - 0.5$ since its PF is a hyperplane that intersects each axis in 0.5. For DTLZ2-DTLZ4 we used $GD = \frac{1}{|P|} \sum_{\mathbf{x} \in P} \|F(\mathbf{x})\|_2 - 1$ since the PF for these problems describes a sphere of radius 1. For DTLZ5-DTLZ7, we used the value of each auxiliary function $g(\mathbf{x})$ defined for each

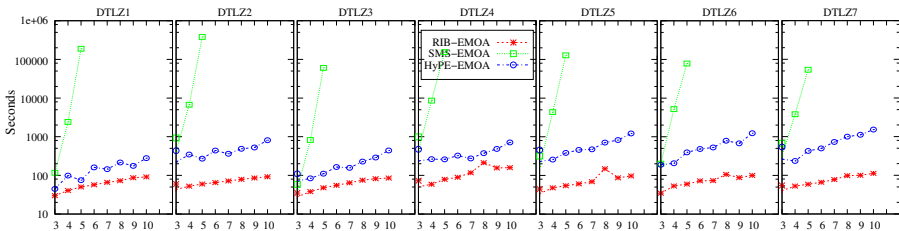


Fig. 2. Average time (axis y) over 30 independent runs for each MOEA when performing 40,000 functions evaluations for each DTLZ test problem (from 3 to 10 objectives functions (axis x))

³ We find out the α value by using the golden search method [12] within the interval (0.05, 20).

problem (for details see [6]). The *PF* of DTLZ5 and DTLZ6 is achieved when $g(\mathbf{x}) = 0$, while the *PF* of DTLZ7 is reached when $g(\mathbf{x}) = 1$. Thus, we used each g function to compute a variant of *GD*, defined by $GD_g = \frac{1}{|P|} \sum_{\mathbf{x} \in P} g(\mathbf{x})$ (for DTLZ5 and DTLZ6) and $GD_g = \frac{1}{|P|} \sum_{\mathbf{x} \in P} g(\mathbf{x}) - 1$ (for DTLZ7), where P denotes the final approximation achieved by each MOEA. Therefore, a value $GD = 0$ indicates that the approximation P is in the real *PF*. For each algorithm, we used $\eta_c = 15$, $\eta_m = 20$, $p_c = 0.9$ and $p_m = 1/n$ for the indexes and ratios in the crossover and mutation operators, respectively. For each MOP,

Table 1. Comparison of results with respect to *GD* for the DTLZ test problems

# Objs.	Algorithm	DTLZ1 <i>GD</i> (σ)	DTLZ2 <i>GD</i> (σ)	DTLZ3 <i>GD</i> (σ)	DTLZ4 <i>GD</i> (σ)	DTLZ5 <i>GD</i> (σ)	DTLZ6 <i>GD</i> (σ)	DTLZ7 <i>GD</i> (σ)
3	RIB-EMOA	0.004213 (0.005947)	0.000041 (0.000048)	17.487000 (7.263660)	0.000029 (0.000045)	0.000001 (0.000003)	3.877700 (0.173200)	0.001201 (0.000394)
	SMS-EMOA	0.001780 (0.001996)	0.000065 (0.000011)	12.160000 (4.633868)	0.000043 (0.000016)	0.000003 (0.000001)	2.617500 (0.067991)	0.001615 (0.000177)
	HyPE-EMOA	0.029672 (0.023956)	0.000458 (0.000071)	12.147000 (5.979108)	0.000363 (0.000165)	0.000040 (0.000011)	4.459300 (0.138107)	0.016254 (0.001617)
4	RIB-EMOA	0.037965 (0.049938)	0.000155 (0.000085)	18.517000 (7.833201)	0.000107 (0.000197)	0.000004 (0.000007)	8.367200 (0.841955)	0.036396 (0.016740)
	SMS-EMOA	0.001861 (0.001276)	0.000201 (0.000025)	17.132000 (8.927643)	0.000115 (0.000047)	0.571280 (0.023577)	3.306000 (0.092850)	0.007434 (0.000678)
	HyPE-EMOA	0.977570 (1.153816)	0.001886 (0.000293)	64.291000 (18.986294)	0.001209 (0.000301)	0.296780 (0.045810)	9.936200 (0.358386)	0.122510 (0.025145)
5	RIB-EMOA	0.029738 (0.027467)	0.000316 (0.000147)	13.776000 (7.468234)	0.000536 (0.000226)	0.004092 (0.009992)	10.489000 (0.639423)	0.157760 (0.045098)
	SMS-EMOA	0.002510 (0.001591)	0.000415 (0.000056)	9.849800 (2.726240)	0.000656 (0.000036)	0.787580 (0.053845)	3.374800 (0.082385)	0.014730 (0.001261)
	HyPE-EMOA	1.929400 (1.138770)	0.005508 (0.000839)	83.570000 (16.132044)	0.003014 (0.000682)	0.330560 (0.043337)	11.866000 (0.346546)	0.307590 (0.059810)
6	RIB-EMOA	0.097693 (0.144339)	0.000809 (0.000255)	20.824000 (11.602267)	0.000887 (0.000321)	0.041967 (0.057430)	12.185000 (0.569374)	0.128210 (0.080949)
	HyPE-EMOA	2.018500 (1.313371)	0.011268 (0.001611)	98.789000 (21.181272)	0.005243 (0.001365)	0.348890 (0.026693)	13.047000 (0.322326)	0.477420 (0.115235)
7	RIB-EMOA	0.374010 (0.525594)	0.001308 (0.000443)	26.081000 (14.997568)	0.001996 (0.000985)	0.065241 (0.103754)	13.460000 (0.480379)	0.122180 (0.041164)
	HyPE-EMOA	2.409300 (1.449388)	0.017296 (0.002415)	99.057000 (19.874297)	0.009122 (0.002152)	0.362480 (0.032717)	13.732000 (0.309724)	0.670290 (0.119358)
8	RIB-EMOA	0.493850 (0.739671)	0.001746 (0.000450)	26.249000 (14.347574)	0.006156 (0.002254)	0.082622 (0.071311)	14.323000 (0.283957)	0.187260 (0.066961)
	HyPE-EMOA	2.451000 (1.522215)	0.023798 (0.003426)	105.050000 (23.295721)	0.017033 (0.004348)	0.380640 (0.036984)	14.494000 (0.247828)	1.020500 (0.090525)
9	RIB-EMOA	0.459380 (0.491440)	0.004350 (0.002952)	26.887000 (16.594691)	0.012242 (0.003688)	0.093066 (0.080599)	14.655000 (0.384037)	0.267120 (0.077010)
	HyPE-EMOA	2.142300 (1.183025)	0.031247 (0.006355)	105.460000 (24.807379)	0.026099 (0.004833)	0.376190 (0.029685)	14.657000 (0.233302)	1.458500 (0.086545)
10	RIB-EMOA	0.312500 (0.366907)	0.008835 (0.004312)	30.731000 (16.845582)	0.018094 (0.006016)	0.118400 (0.097462)	15.062000 (0.371148)	0.446840 (0.126770)
	HyPE-EMOA	2.574600 (1.366155)	0.034142 (0.005567)	109.580000 (19.801898)	0.039354 (0.010316)	0.398900 (0.026573)	15.090000 (0.297923)	1.962400 (0.086160)

30 independent runs were performed with each algorithm. We employed a population size $N = 200$ and the search was restricted to 40,000 fitness function evaluations for each problem. The cardinality of the reference set was set as $\mu = \rho \times N$ (here, we used $\rho = 3$). The results obtained are summarized in Table 1. This table displays both the *average* and the standard deviation (σ) for the *GD* performance measure for each MOP. For an easier interpretation, the best results are presented in **boldface**.

In our study, we tested the abilities of RIB-EMOA using our proposed reference set construction when solving MOPs with many objectives (between three and ten objective functions). From Table 1, we can see that RIB-EMOA obtained better approximations to the real *PF* than HyPE-EMOA in most of the test problems. Nevertheless, SMS-EMOA obtained better results than RIB-EMOA for DTLZ1, DTLZ3 and DTLZ6 test problems. The poor performance of RIB-EMOA in these problems could be due to the high multi-modality (in the case of DTLZ1 and DTLZ3) and the degeneration (in the case of DTLZ6) that these problems have in their *PF*s. Although DTLZ5 also has a degenerate *PF*, it is much more difficult to approximate solutions to the real *PF* of DTLZ6 than to the real *PF* of DTLZ5 (for details of these problems see [6]). In fact, RIB-EMOA relies on the proper construction of the reference set which is constructed from the individual minima of each problem. Thus, given the features of these MOPs, the achievement function (in equation (8)) which establishes the individual minima could be not the best in order to construct a proper surface for them. Nonetheless, an improvement mechanism for our proposed reference set construction, is indeed, a possible path for future research. On the other hand, according to Fig. 2, we can see that SMS-EMOA achieved good results for DTLZ1, DTLZ3 and DTLZ6 (see Table 1) but consuming a higher computational time than RIB-EMOA. However, the computational time required by our RIB-EMOA was lower than that of SMS-EMOA and HyPE-EMOA even when solving MOPs with 10 objective functions. Moreover, the time consumption for SMS-EMOA was so high that we could only test it with MOPs having up to 5 objectives. Based on the previous discussion, we consider that our proposed approach is a good choice in order to deal with MOPs with a high number of objectives.

6 Conclusions and Future Work

In this paper, we have presented a first attempt to generalize a reference set for a given MOP. For this sake, we have considered the fact that the *PF* of a MOP could be described as a linear, convex, or concave manifold in the objective function space. In such cases, the *PF*s present geometries which can be associated to a curve of the family described in equation (4). The proposed reference set construction was found to be appropriate, since it yields a suitable surface in order to approximate solutions to the real *PF* by using the Δ_p indicator (however, it could also be used with other MOEAs that adopt a reference set). According to our results, we showed the potential of our proposed approach for attracting solutions towards the real *PF* along the search process. It is indeed desirable to compare our proposed approach against more state-of-the-art MOEAs and this will be part of our future work. It is worth noting that our RIB-EMOA produced competitive results even with respect to problems having non-well-defined *PF*s (e.g., in DTLZ5-DTLZ7 (these MOPs have discontinuities and degenerations in their *PF*s)), which could be a useful feature when dealing with real-world MOPs.

As part of our future research, we intend to focus on designing another strategy in order to improve the construction of the reference set. It is also desirable to introduce the use of preferences to our proposed approach. Finally, we also aim to extend our proposed approach to deal with constrained MOPs having many objectives, which is an area that has remained practically unexplored so far, to the authors' best knowledge.

References

1. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. *EJOR* 181(3), 1653–1669 (2007)
2. Brockhoff, D., Wagner, T., Trautmann, H.: On the properties of the R2 indicator. In: *GECCO 2012*, pp. 465–472. ACM (2012)
3. Coello Coello, C.A., Cruz Cortés, N.: Solving Multiobjective Optimization Problems using an Artificial Immune System. *Genetic Programming and Evolvable Machines* 6(2), 163–190 (2005)
4. Das, I.: Nonlinear Multicriteria Optimization and Robust Optimality. PhD thesis, Rice University, Houston, Texas (1997)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE TEVC* 6(2), 182–197 (2002)
6. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Ajith, A., et al. (eds.) *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pp. 105–145. Springer, USA (2005)
7. Diaz-Manriquez, A., Toscano-Pulido, G., Coello Coello, C.A., Landa-Becerra, R.: A ranking method based on the R2 indicator for many-objective optimization. In: *CEC 2013*, pp. 1523–1530. IEEE (2013)
8. Dominguez-Medina, C., Rudolph, G., Schütze, O., Trautmann, H.: Evenly spaced pareto fronts of quad-objective problems using psa partitioning technique. In: *CEC 2013*, pp. 3190–3197. IEEE (2013)
9. Fang, K.T.: The Uniform Design: Application of Number-Theoretic Methods in Experimental Design. *Acta Math. Appl. Sinica* 3, 363–372 (1980)
10. Hansen, M.P., Jaszkiewicz, A.: Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Institute of Mathematical Modeling, Technical University of Denmark (1998)
11. Hernandez Gomez, R., Coello Coello, C.A.: MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator. In: *CEC 2013*, pp. 2488–2495. IEEE (2013)
12. Kiefer, J.: Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society* 4(3), 502–506 (1953)
13. Rodríguez Villalobos, C.A., Coello Coello, C.A.: A new multi-objective evolutionary algorithm based on a performance assessment indicator. In: *GECCO 2012*, pp. 505–512. ACM (2012)
14. Schütze, O., Esquivel, X., Lara, A., Coello Coello, C.A.: Using the averaged Hausdorff distance as a performance measure in evolutionary multi-objective optimization. *IEEE TEVC* 16(4), 504–522 (2012)
15. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio (May 1999)
16. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) *PPSN 2004*. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)

Travelling Salesman Problem Solved ‘*in materio*’ by Evolved Carbon Nanotube Device

Kester Dean Clegg¹, Julian Francis Miller¹, Kieran Massey², and Mike Petty²

¹ The University of York, UK

{kester.clegg, julian.miller}@york.ac.uk

² Durham University, UK

{m.k.massey, m.c.petty}@durham.ac.uk

Abstract. We report for the first time on finding shortest path solutions for the travelling salesman problem (TSP) using hybrid “*in materio*” computation: a technique that uses search algorithms to configure materials for computation. A single-walled carbon nanotube (SWCNT) / polymer composite material deposited on a micro-electrode array is configured using static voltages so that voltage output readings determine the path order in which to visit cities in a TSP. Our initial results suggest that the hybrid computation with the SWCNT material is able to solve small instances of the TSP as efficiently as a comparable evolutionary search algorithm performing the same computation in software. Interestingly the results indicate that the hybrid system’s search performance on TSPs scales linearly rather than exponentially on these smaller instances. This exploratory work represents the first step towards building SWCNT-based electrode arrays in parallel so that they can solve much larger problems.

1 Introduction

As material variability starts to limit the design of modern silicon-based processors [1,2], the NASCENCE project [3] has started investigating how nano-scale materials and devices that could be configured for computation [4]. While engineering for computation at such small scales remains extremely difficult, it is possible to use a method of automatically configuring materials so that they can perform a dedicated task [5]. In this paper, we illustrate a method to achieve this using a single-walled carbon nanotube (SWCNT) material configured by applying static voltages to specific locations on a micro-electrode array. The remaining electrodes act as inputs to the computation, in this case a small instance of the Travelling Salesman Problem (TSP). The hybrid system successfully finds the shortest path in many cases and its search performance is on a par to a comparable (optimised) evolutionary algorithm implemented in software. We believe this work represents the first time a benchmark search problem has been solved by a machine configured material and could pave the way toward designing novel computational materials: resulting in small, single purpose devices that are stable, low voltage and yet able to compute answers to challenging tasks.

Previous work in computer controlled configuration of physical systems for computation has had limited success [6,7,8]; results were sometimes unstable [9,10], able to

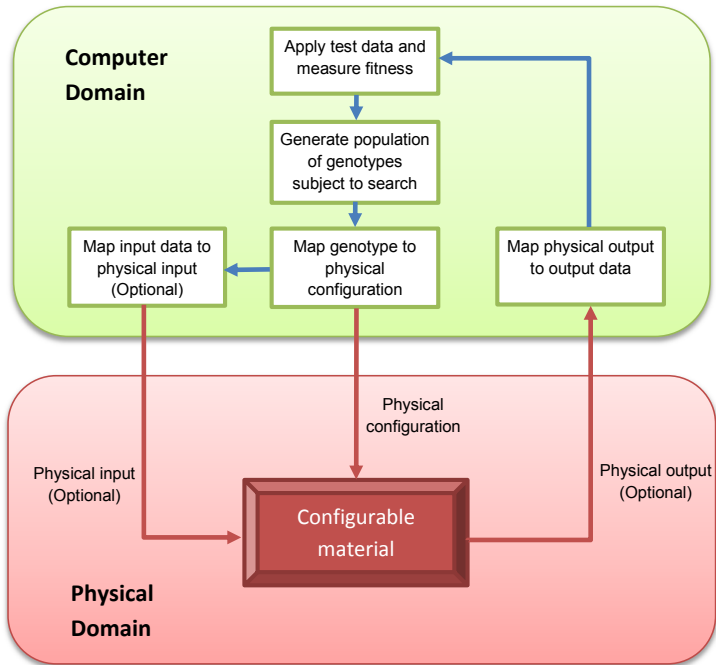


Fig. 1. Overview of hybrid computation with ‘in materio’ configuration performed by evolutionary search. The computer configures the application of physical signals to a material and tests the output. A genotype of configuration instructions is subject to evolutionary mutation. Physical output from the materials acts as computational input to the selection process. The configuration loop stops when the computational inputs have found a solution or no more improvement is required or possible.

carry out only low level processing tasks [6,11] or performed no better than exhaustive search [12,13]. Most work done on configuring materials to carry out computation has used search-based algorithms to exploit physical characteristics within the material without much concern as to how those characteristics could be formally understood or engineered [14,15]. An overview of the hybrid approach to materials-based computation is shown in Figure 1.

2 Hardware and Material Description

The NASCENCE consortium [3] is investigating candidate materials and techniques that could be employed to configure materials for computation. The project is looking at SWCNT held in fixed, gel and liquid mediums (such as liquid crystal). In this paper we report results using a fixed material system (supplied by Durham University) prepared on gold micro-electrode arrays with contacts arranged in either a 3x4 or a 4x4 grid. The early prototype 3x4 grid array has pads that are 40 μ m diameter with a pitch of

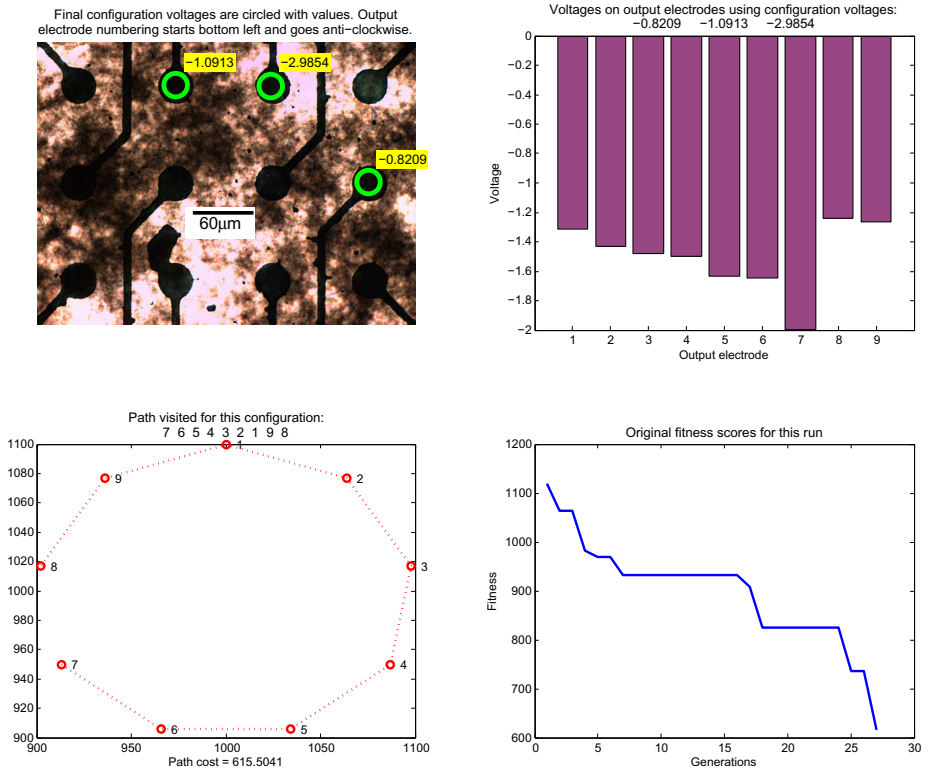


Fig. 2. Top left: shows the CNT dispersal over an earlier prototype 3x4 grid electrodes (x200 mag). Note unevenness of material over electrodes and the mask fault on the third electrode does not appear to affect the evolutionary search when finding the shortest path of the TSP (the evolutionary history is shown by the best performing genotype for each generation, bottom right). In this final configuration, voltages are applied to the circled electrodes and the remaining electrodes provide the floating point values into the TSP. Top right: recorded voltages which when sorted determine the path order to visit cities. Bottom left: Optimum path solution of the TSP.

100µm, leaving an electrode separation of approximately 60µm, while the later 4x4 grid has a spacing of 50µm pads and separation. The material is composed of 0.10% (by weight) of single-walled carbon nanotubes (purchased from Carbon Nanotechnologies Inc. Houston, TX, USA) mixed with poly-methyl methacrylate (PMMA, Aldrich Mw = 93,000) and dissolved in Anisole (VWR, analytical reagent grade). Approximately 20µL of material is dispensed on the electrode array which is then dried at 100°C for 30mins to leave a “thick film”. The density of SWCNTs within the PMMA is quite variable, as is the accuracy of the mask to create the electrodes (see top left panel, Figure 2), however these inconsistencies in the substrate do not appear to present a problem to evolutionary search.

The 4x4 electrode array is connected a 16x16 analogue cross point switch which controls connections from the substrate to a data acquisition (DAQ) card, enabling us

to place configuration voltages anywhere within the array. The DAQ card first digitally configures the switch connections and then inputs analogue configuration voltages to the material and records the corresponding analogue outputs. The number of configuration voltages deployed depends on the problem being tackled and the availability of spare electrodes on the array. The configuration voltages and electrodes to which they connect are decided by a 1+4 evolutionary algorithm (see method). The range is restricted to $\pm 3\text{V}$ and all connections are one to one (i.e. one configuration voltage can only go to one electrode). We apply the voltages for 1s and take mean values of the final 0.2s to minimize any noise or “settling periods” within the material. The time required to configure the analogue switch and set up channels on the DAQ card means that testing a configuration takes several seconds. While this is relatively “slow” to perform a computation, signals from the SWCNT-PMMA materials have negligible noise levels after the initial 50ms and our sampling times could be substantially reduced.

3 Problem and Method

The TSP was selected as our computational task as the problem belongs to a class of computationally hard problems known as NP-complete, it has been extensively studied [16] and there exist excellent free solvers against which we can check our solutions¹. In terms of the search, the total number of paths within a given TSP increases factorially with the number of cities to visit, making it an excellent problem to see whether a solver can scale to larger problems.

A circular layout of cities has an easily identifiable shortest path. For symmetric TSPs, the minimum number of shortest paths is equal to double the number of cities, as a path can start at any city and go in either direction along the shortest path. Thus a circular 9 city TSP has 18 shortest paths, while an irregular 9 city layout may have more. A path can be expressed as a permutation, therefore the total number of paths for a given TSP is given by the factorial of the number of cities:

9 city	= 9!	= 362,880
10 city	= 10!	= 3,628,800
11 city	= 11!	= 39,916,800

Our representation maps the path visiting each city to a sorted vector of floating point numbers (the voltage outputs from the substrate, see below). The computational aim is to get the evolutionary algorithm to select the connections and configure the material with voltages so that it outputs a vector that represents the shortest path in the TSP (see bottom left of Figure 2). The sorted vector, or ‘smallest position value’ representation [17], can be readily applied to many other well-known computational problems such as classification tasks [18].

Currently, when we tackle a 10 city TSP with a 4x4 electrode array, we can in theory use the remaining 6 voltages to configure the SWCNT-PMMA material. In practice we are limited to a maximum of 4 analogue outputs from our current DAQ card and the results presented here reflect this limitation. However, we are having bespoke hardware

¹ <http://www.math.uwaterloo.ca/tsp/concorde/>

built for the NASCENCE consortium [3] that will allow many more analogue inputs and outputs to connect to the electrode arrays and permit us to look at a broader range of problems.

For example, using our current electrode array and DAQ card to configure the SWCNT-PMMA material on a 12 city TSP, we can use between 2–4 configuration voltages to configure the substrate and use the remaining electrodes of the array to provide the 12 input values to the problem. The input vector of values (1.a below — note just a few values are shown for clarity) is converted to a path by using the original indices after sorting (1.b); the sorted indices then become the order in which to visit the cities (see also bottom left Figure 2). In the following example city 6 would be visited first, then 2, then 4 and so on:

1. a)	1.892	0.321	2.064	0.984	1.664	0.056	2.391	2.907
	1	2	3	4	5	6	7	8
1. b)	0.056	0.321	0.984	1.664	1.892	2.064	2.391	2.907
	6	2	4	5	1	3	7	8

The use of a 16x16 cross point switch permits an evolutionary algorithm to select which electrodes in the array will be used for the configuration voltages and which are to be used as inputs to the TSP. This selection allows the search algorithm to exploit the physical resources available to different electrodes (this can be important, note the variable SWCNT dispersion shown top left of Figure 2). After the path order to visit cities is determined, the path length is calculated and this represents the evolutionary fitness for that configuration.

The evolutionary algorithm uses a 1+4 schema, where the best is kept and 4 copies are mutated for each generation. If a mutant scores an equivalent fitness to the previous best it is selected to form the subsequent generation so that the search continues to ‘move on’ despite not increasing solution fitness. The schema was chosen for its simplicity rather than its efficiency [19,20,21]. We impose a limit of 1500 generations on the search, as a run that fails to find the shortest path within this will have taken approximately 14 hours to complete. However, failure to solve these smaller instances of the TSP using the SWCNT-PMMA material is relatively rare (although the failure rate has been observed to increase on larger problems with fewer configurations); so allowing the search to continue beyond 1500 generations seems unlikely to impact the average number of generations to find the shortest path.

4 Results

To the best of our knowledge, this work represents the first time the TSP has been solved using search algorithms to configure a material for computation. Knowing the exact size of the solution space for small instances of TSP allows us to calculate the proportion of samples taken from the total solution space that were required to find the shortest path. By comparing the number of samples taken from the total solution space to the number taken by doing the same computation entirely in software, we get an insight into whether performing the computation via the SWCNT-PMMA material has any advantages over doing the same task entirely in software. In order to do this, we wanted to chose an evolutionary algorithm that could be meaningfully compared to

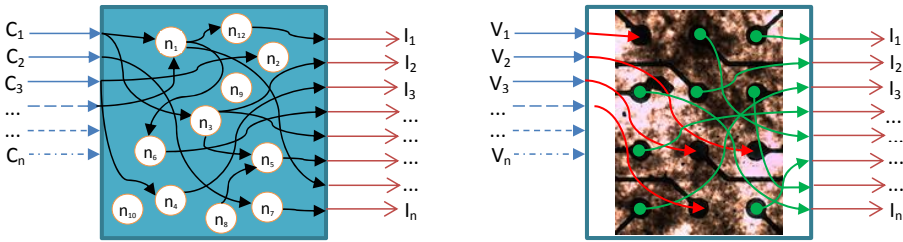


Fig. 3. The two evolutionary search algorithms are not exact correlates. The LHS represents the CGP evolved network of nodes and mathematical operators. In this schema, the inputs $C_{1..n}$ to the network are constants initially chosen at random but which do not change. The remaining connections and operators within each node are open to mutation, including those nodes selected as connecting to the inputs $I_{1..n}$ to the TSP. The RHS shows the SWCNT-PMMA material over an early 4x3 electrode. In this case the evolutionary algorithm is able to select which electrodes serve as inputs to the TSP and which will take the configuration voltages $V_{1..n}$ (note that in contrast to CGP, the input voltages are subject to evolutionary selection).

our hybrid material-based computation. Although we have no current means to analyse what electrical networks or properties are created within the SWCNT-PMMA material during its configuration, we can treat it as a ‘black box’ which takes certain inputs and transforms them into our desired outputs.

Cartesian Genetic Programming (CGP [22]) is a technique that creates a similar ‘black box’ of up to a 1000 networked nodes, within which mathematical and arithmetic operators transform inputs before passing on the values to other nodes or outputs [23]. The evolutionary mutation of the genotypes that describe the CGP network has parallels with our evolutionary mutation of the voltages and switch connections that affect the SWCNT-PMMA material’s configuration. We show a schematic comparison of the two methods in Figure 3. As CGP is known to be an efficient technique at solving well known benchmark problems [24] one would not normally expect a 1+4 evolutionary schema in software to outperform a similar schema in CGP. We encoded the same TSP instances to be solved by CGP and compared its performance against the SWCNT-PMMA material. It should be noted that as CGP running in software runs many times quicker than the hybrid SWCNT-PMMA system, it was possible to experiment with the CGP parameters to optimise its performance on the problem. Indeed, we discovered that we could improve the average number of generations to find the shortest path for CGP by a factor of ten or more. Time constraints meant this sort of experimentation was impossible for the SWCNT-PMMA material.

Since Thompson’s work it has been known that evolutionary algorithms with access to physical resources can make use of them in ways that are difficult to analyse or even predict [25]. Even so we were not expecting that instances of the TSP would be solved almost as efficiently (in terms of sampling the solution space) and reliably (in terms of successful runs) by the hybrid system using the SWCNT-PMMA material as an optimised encoding of CGP. While it is difficult to draw exact comparisons between the methods, both Table 1 and Figure 4 show that increasing the number of configuration

Table 1. Results for an optimised version of CGP and the hybrid evolutionary system with SWCNT-PMMA material for different instances of the TSP. Note proximity in the median number of generations of highlighted instances despite a decrease in the average number of samples from the solution space by a factor of almost ten.

Computation type	Size of TSP	No. of configuration voltages	Average no. of generations for successful runs	Median no. of generations	Average % sample of solution space
SWCNT-PMMA substrate	9	2	158.6	104.5	0.1751
	9	3	57.36	42.5	0.0741
	9	4	118.4	61.5	0.1308
	10	2	157.95	155	0.0174
	10	3	79.76	63.5	0.0086
	10	4	68.03	46.5	0.0075
	11	2			
	11	3			
	11	4	170.7	92	0.0017
Software (CGP encording)	9	n/a	34.04	29.5	0.4599
	10	n/a	48.96	40.5	0.0643
	11	n/a	91.96	65.5	0.0065

voltages exposed to evolutionary selection seems to improve search performance up to point, beyond which the performance starts to degrade again. We do not yet have a hypothesis as to why this is as it is impossible to know what physical characteristics in the material that the search algorithm is exploiting. However, the results hold true across different SWCNT-PMMA materials (i.e. changing the percentage of CNT within the PMMA) and across different city layouts for the TSP instances (e.g. random and irregular geometric arrangements).

Some instability observed in the population's fitness scores during the evolutionary runs for TSPs with 2 configuration voltages (first column in Figure 4) led us to suspect that increasing the number of configuration voltages on larger instances would lead to better performance. While we observed almost identical levels of performance for 10 city with 4 configuration voltages as for 9 city with 3, we are unable to verify the performance of the hybrid system on an 11 city TSP beyond 4 configuration voltages as this is the maximum number of analogue outputs that our current equipment supports. It may be the case that there is a "sweet spot" for optimal performance based on the proportion of configuration voltages to the number of cities in the TSP instance, which from our (limited) initial results might be somewhere between a third and a half (e.g. 12 city might need around 5).

The instability during some evolutionary runs is caused by voltages that are recorded as floating point numbers in the sorting vector converging to nearly identical values. Noise (in the μV range) can then cause a radical difference to the permutation that determines which order the cities are visited. This re-ordering of vector values can mean that a search run which had reached a plateau representing a local fitness optima can escape and go on to find the global optimum. We suspect this "feature" of the hardware may be one factor that helps the SWCNT-PMMA materials outperform CGP in one case, although we are aware this argument is contrary to results for the 9 city TSP with 3 and 10 city with 4 configuration voltages as in these instances there is almost no instability recorded during the evolutionary runs and there may well be more important factors.

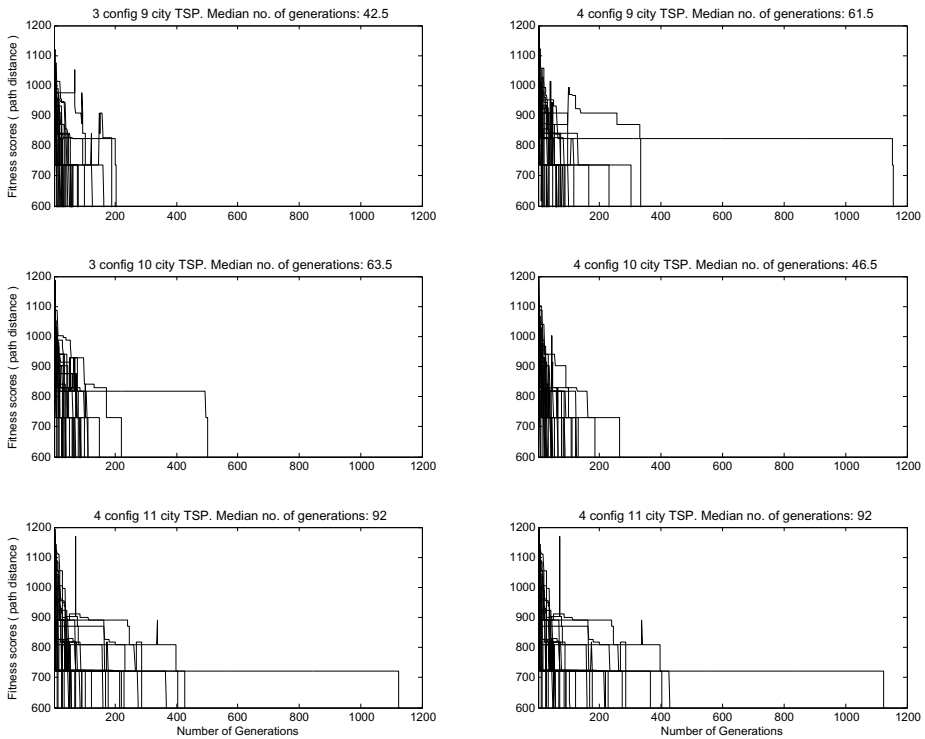


Fig. 4. The plots show the evolutionary histories of 30 runs of 9, 10 and 11 TSP instances using the same SWCNT-PMMA material over 1500 generations. There appears to be a “sweet spot” for the number of configurations voltages for optimal search performance that may increase as the number of cities increases. Unfortunately we cannot confirm this as our current hardware is limited to a maximum of 4 configuration voltages. Note how the material struggles with insufficient configuration voltages; those runs using 2 configuration voltages all show instability and occasional failure to find the shortest path. The median number of generations to find the shortest path is given as some runs have outliers.

Another intriguing finding is that despite the factorial increase in solution space between 9, 10 and 11 city TSPs, the average number of generations for the SWCNT-PMMA material to find the shortest path remains almost constant in the best cases. This might suggest that the performance of the hybrid system with the SWCNT-PMMA material could have a linear time complexity (rather than exponential) on NP complete problems such as the TSP. However until we are able to build and test larger SWCNT-PMMA electrode arrays, we cannot confirm that this is the case and can only say that the potential for the hybrid SWCNT-PMMA system to solve large instances of the TSP efficiently appears promising.

5 Conclusion

Currently our 16 electrode array and low number of analogue outputs restrict the maximum size of TSP we can tackle. However, our most recent experiments suggest it is possible to split the genome of TSP configuration so that the problem is effectively halved. This allows us to sequentially process much larger TSP instances using the same array. While this appears to work, the evolutionary runs are more than doubled in duration making it difficult to acquire meaningful performance data. However, as a proof of concept it lays the way towards trying to tackle scalability by making two 4x4 arrays and using them to solve a single larger instance of the TSP. Such an approach would demonstrate parallel computation that could massively speed up search times, as all inputs to the problem are read simultaneously. While we are at an early stage in this research, we believe that it may be possible in the future to configure many thousands of small SWCNT-based electrode arrays to perform a computation in the way we currently manufacture and program millions of logic gates in silicon.

References

1. Walker, J., Trefzer, M., Bale, S., Tyrrell, A.: Panda: A reconfigurable architecture that adapts to physical substrate variations. *IEEE Transactions on Computers* 62(8), 1584–1596 (2013)
2. Walker, J.A., Trefzer, M.A., Tyrrell, A.M.: Designing function configuration decoders for the PANDA architecture using multi-objective cartesian genetic programming. In: Suganthan, P.N. (ed.) 2013 IEEE Symposium Series on Computational Intelligence, Singapore, April 16–19, pp. 96–103 (2013)
3. Broersma, H., Gomez, F., Miller, J.F., Petty, M., Tufte, G.: Nascence project: Nanoscale engineering for novel computation using evolution. *International Journal of Unconventional Computing* 8(4), 313–317 (2012)
4. Graham, A.P., Duesberg, G.S., Hoenlein, W., Kreupl, F., Liebau, M., Martin, R., Rajasekharan, B., Pamler, W., Seidel, R., Steinhögl, W., Unger, E.: How do carbon nanotubes fit into the semiconductor roadmap? *Applied Physics A* 80(6), 1141–1151 (2005)
5. Harding, S., Miller, J.F.: Evolution in materio: Evolving logic gates in liquid crystal. *International Journal of Unconventional Computing* 3(4), 243–257 (2007)
6. Thompson, A.: Evolving electronic robot controllers that exploit hardware resources. In: Morán, F., Merelo, J.J., Moreno, A., Chacon, P. (eds.) ECAL 1995. LNCS, vol. 929, pp. 640–656. Springer, Heidelberg (1995)
7. Miller, J.F., Harding, S.L., Tufte, G.: Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence* 7, 49–67 (2014)
8. Mills, J.W.: Polymer processors, tech. rep. tr580. Technical report, Department of Computer Science, University of Indiana (1995)
9. Harding, S., Miller, J.F.: Evolution in materio: Investigating the stability of robot controllers evolved in liquid crystal. In: Moreno, J.M., Madrenas, J., Cosp, J. (eds.) ICES 2005. LNCS, vol. 3637, pp. 155–164. Springer, Heidelberg (2005)
10. Thompson, A.: Evolving fault tolerant systems. In: Proc. 1st IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 1995), IEE Conf. Publication No. 414, pp. 524–529 (1995)
11. Harding, S., Miller, J.: Evolution in materio: a tone discriminator in liquid crystal. In: Congress on Evolutionary Computation, CEC 2004, vol. 2, pp. 1800–1807 (June 2004)

12. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. *Science* 266 (5187), 1021–1024 (1994)
13. Adamatzky, A.: *Reaction-Diffusion Automata: Phenomenology, Localisations, Computation*. Springer, Heidelberg (2013)
14. Thompson, A.: Exploring beyond the scope of human design: Automatic generation of FPGA configurations through artificial evolution (Keynote). In: *Proc. 8th Annual Advanced PLD & FPGA Conference*, Miller Freeman, pp. 5–8 (1998)
15. Harding, S.L., Miller, J.F., Rietman, E.A.: Evolution in materio: Exploiting the physics of materials for computation. *International Journal of Unconventional Computing* 4(2), 155–194 (2008)
16. Reinelt, G.: *The Traveling Salesman*. LNCS, vol. 840. Springer, Heidelberg (1994)
17. Fatih Tasgetiren, M., Sevkli, M., Yun-Chia, L., Gencyilmaz, G.: Particle swarm optimization algorithm for single machine total weighted tardiness problem. In: *Congress on Evolutionary Computation, CEC 2004*, vol. 2, pp. 1412–1419 (2004)
18. Tasgetiren, F., Chen, A., Gencyilmaz, G., Gattoufi, S.: Smallest position value approach. In: Onwubolu, G.C., Davendra, D. (eds.) *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*. SCI, vol. 175, pp. 121–138. Springer, Heidelberg (2009)
19. Rechenberg, I.: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog (1973)
20. Schwefel, H.P.: *Numerical optimization of Computer models*. John Wiley & Sons, Ltd. (1981)
21. Beyer, H.G., Schwefel, H.P., Wegener, I.: How to analyse evolutionary algorithms. *Theoretical Computer Science* 287(1), 101–130 (2002)
22. Miller, J.F.: *Cartesian Genetic Programming*. Springer (2011)
23. Miller, J.F., Mohid, M.: Function optimization using Cartesian Genetic Programming. In: *Proc. Conf. on Genetic and Evolutionary Computation (Companion)*, pp. 147–148 (2013)
24. Walker, J., Miller, J.F.: The automatic acquisition, evolution and re-use of modules in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation* 12, 397–417 (2008)
25. Thompson, A.: An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. In: Higuchi, T., Iwata, M., Weixin, L. (eds.) *ICES 1996*. LNCS, vol. 1259, pp. 390–405. Springer, Heidelberg (1997)

Randomized Parameter Settings for Heterogeneous Workers in a Pool-Based Evolutionary Algorithm

Mario García-Valdez¹, Leonardo Trujillo¹,
Juan Julián Merelo-Guérvos², and Francisco Fernández-de-Vega³

¹ Instituto Tecnológico de Tijuana, Tijuana B.C., México

² Universidad de Granada, Granada, Spain

³ Grupo de Evolución Artificial, Universidad de Extremadura, Mérida, Spain
{mario, leonardo.trujillo}@tectijuana.edu.mx
jmerelo@geneura.ugr.es, fcofdez@unex.es

Abstract. Recently, several Pool-based Evolutionary Algorithms (PEAs) have been proposed, that asynchronously distribute an evolutionary search among heterogeneous devices, using controlled nodes and nodes outside the local network, through web browsers or cloud services. In PEAs, the population is stored in a shared pool, while distributed processes called workers execute the actual evolutionary search. This approach allows researchers to use low cost computational power that might not be available otherwise. On the other hand, it introduces the challenge of leveraging the computing power of heterogeneous and unreliable resources. The heterogeneity of the system suggests that using a heterogeneous parametrization might be a better option, so the goal of this work is to test such a scheme. In particular, this paper evaluates the strategy proposed by Gong and Fukunaga for the Island-Model, which assigns random control parameter values to each worker. Experiments were conducted to assess the viability of this strategy on pool-based EAs using benchmark problems and the EvoSpace framework. The results suggest that the approach can yield results which are competitive with other parametrization approaches, without requiring any form of experimental tuning.

Keywords: Pool-based Evolutionary Algorithms, Distributed Evolutionary Algorithms, Algorithm Parametrization.

1 Introduction

Evolutionary computation (EC) research has allowed scientists and engineers from many fields to understand the power of the natural search process described by the biological theory of Neo-Darwinian evolution [13]. Inspired in biological evolution, EC researchers have developed a variety of search and optimization algorithms [6]. While EAs are inspired by evolution, they mostly follow an abstract model of the natural process. For instance, one aspect that is omitted from most EAs is the open-ended nature of evolution, in practice EAs are used to solve problems with well defined objectives, while natural evolution is an adaptive process without an a priori goal or purpose. Biological evolution is also an intrinsically parallel, distributed and asynchronous process,

undoubtedly important features that have allowed evolution to produce impressive results throughout nature. However, some of these features are not trivially included into standard EAs [1], which are mostly coded as sequential and synchronous algorithms [6]. For instance, a large body of work exists in EA parallelization, a comprehensive introduction can be found in [1]. However, distributed and asynchronous EAs have started to become common only recently. In particular, recent trends in information technology have opened new lines of future development for EC research.

Today, computing resources range from personal computers and smart-devices to massive data centers. These resources are easily accessible through Internet technologies, such as cloud computing, peer-to-peer networks and web environments. Several EAs have been proposed that distribute the evolutionary process among heterogeneous devices, not only among controlled nodes within an in-house cluster or grid, but also to others outside the data center, in web browsers, smart phones or cloud services. This reach out approach allows researchers to use low cost computational power that would not be available otherwise, but on the other hand, have the challenge to manage heterogeneous and mostly unreliable computing resources. In particular, we are interested in systems that follow a pool-based approach, where the search process is conducted by a collection, of possibly heterogeneous, collaborating processes using a shared repository or population pool. We will refer to such algorithms as pool-based EAs or PEAs, which are intrinsically parallel, distributed and asynchronous.

Despite promising results, PEAs present several challenges. From a technological perspective, lost connections, low bandwidth, abandoned work, security and privacy are all important issues. This work, however, focuses on algorithm parametrization, a common issue with most EAs that is amplified in a PEA. In general, EAs are sometimes criticized by the large number of parameters they possess, that for real world problems need to be tuned empirically or require additional heuristic processes to be included into the search to adjust the parameters automatically [15, 16]. In the case of a PEAs, this issue is magnified since the underlying system architecture adds several degrees of freedom to the search process, with unknown interactions.

This work studies the recently proposed EvoSpace system, a framework to develop PEAs using an heterogeneous collection of possibly unreliable computing resources. Despite promising initial results [9–11, 21], research devoted to EvoSpace has not addressed the parametrization issue. Therefore, in this paper the recent approach called Randomized Parameter Setting Strategy (RPSS) [12, 20] is tested with EvoSpace. The idea behind RPSS is that in a distributed EA, algorithm parametrization may be completely skipped and still conduct a successful search. Results suggest that when the number of distributed process is large enough, algorithm parameters can be set randomly and still achieve good results. However, work on RPSS has only focused on the well-known Island Model for EAs, a distributed but synchronous system. On the other hand, the goal of this work is to evaluate RPSS on a PEA implemented through EvoSpace, that does not have a fixed population structure.

The remainder of the paper proceeds as follows. Section 2 reviews related work. Section 3 briefly describes the EvoSpace framework and provides implementation details. The problem statement and experimental work are presented in Section 4. Finally, concluding remarks are outlined in Section 5.

2 Related Work

In terms of parallelizing EAs, a large body of work has been developed, covering many different EAs, with important practical and theoretical results [1]. However, only recently has the topic of distributed systems been explored. For instance, Fernández et al. [8] use the Berkeley Open Infrastructure for Network Computing (BOINC) to distribute EA runs across an heterogeneous network of volunteer computers using virtual machines. However, such a system, as well as most distributed and parallel systems, does not follow the PEA approach studied in the current paper. In general, a pool-based system employs a central repository where the evolving population is stored. Distributed clients interact with the pool, performing some or all of the basic EA processes (selection, genetic operators, survival). For example, Smaoui Feki et al. [19] uses BOINC redundancy to deal with the volatility of computing nodes. In that work each BOINC work unit consisted of a fitness evaluation task and multiple replicas were produced and sent to different clients. Merelo et al. [17] developed a Javascript PEA that distributes the evolutionary process over web browsers, this provides the added advantage of not requiring additional software installations on client machines. Other similar cloud-based solutions are based on a global queue of tasks and a Map-Reduce implementation [5, 7, 18]. The current work, however, focuses on the EvoSpace presented in [9–11, 21] and summarized in Section 3.

One of the main problems with implementing successful EAs is parameter tuning [16], of particular importance in real-world scenarios, where usually there is little prior insights regarding what might be the best configuration for an EA, especially if the intent is to use it as a black-box optimizer; a comprehensive survey on this topic is given in [16]. Indeed, this problem has received a growing level of interest in recent years, as evidenced by the Self-Search track at GECCO for instance, where the aim is to develop self-adaptive or auto-tuning systems that reduce the amount of human intervention that might be required before performing an EA-based search or optimization.

A noteworthy contribution in [16] is the chapter by Cantú Paz that tackles the problem of deriving theoretical models of the effects of key EA parameters, such as population size and migration schemes in Island-Model EAs (IMEA) [2]. However, it does not cover the effects of all possible parameters, or the specific intricacies of a PEA algorithm. Here, we would stress some important differences between PEAs and IMEAs. First, an IMEA presents a fixed topological structure, with a predefined interaction protocol among each evolving population, this leads to a coordinated, or even synchronized, interaction between the islands. On the other hand, a PEA does not include such constraints, which means that the interactions between workers is much less structured or controlled. Second, in an IMEA each island represents an individual evolutionary process, sharing some of the same dynamics as standard EAs. In a PEA, however, only a single centralized population exists, samples of which are distributed across workers, but ultimately combined once again in the centralized pool. Therefore, some of the well-known insights derived from IMEA research (regarding, for example, migration policies) are not necessarily relevant in the PEA framework. Therefore, new parametrization approaches must be explored.

Several parameter tuning methodologies for EAs are presented in [16], that can substantially reduce the computational cost when compared with an exhaustive search in

parameter space. However, these methods focus on standard EAs, based on serial and synchronous searchers. On the other hand, recent works [12, 20] have shown a promising simpler alternative to tune EAs that employ multiple evolving populations. In particular, the RPSS approach proposed in [12] which quite intriguing given its simplicity. First, consider the original configuration studied in [12, 20], an IMEA where a set of N separate populations, or demes, run semi-isolated evolutionary processes, organized using a particular neighborhood structure, such as ring or a random graph. Each deme is not totally isolated, since after a certain amount of time (generations or function evaluation) a set of individuals is exchanged between neighboring demes, a process known as migration. Obviously, drastically increasing the number of system parameters makes sweeping parameter space computationally unfeasible, since it is not possible to assume that the best configuration is an homogeneous system where all demes share the same parametrization. Moreover, the additional complexity of the Island Model incorporates additional degrees of freedom that must be tuned before performing a run. Such a tuning task can become overwhelming, particularly if the number of islands is large. Therefore, the proposal in [12] is to set the parameter values randomly, without a tuning or self-adaptive process whatsoever. The RPSS approach is to set the parameters of each deme randomly at the beginning of the run, a very simple and apparently naive approach. Nevertheless, results reported in [12, 20] show promise, achieving competitive results while substantially reducing the amount of effort required to tune the system (the approach only requires the user to specify a range of valid values for each parameter).

3 EvoSpace

EvoSpace is based on the tuple space model [11], consisting of two main components (see figure 1): (i) the EvoSpace container that stores the population and (ii) EvoWorkers, which execute the actual evolutionary process, while EvoSpace acts only as a population repository. In a basic configuration, EvoWorkers pull a small random subset of the population, and use it as the initial population for a local EA executed on the client machine. Afterward, the evolved population from each EvoWorker is returned to the EvoSpace container. When individuals are pulled from the container they remain in a phantom state, they cannot be pulled again but they are not deleted. Only if the EvoWorker returns new individuals, are the phantom individuals deleted. If the EvoSpace container is at risk of starvation or when a time-out occurs, phantom individuals are reinserted and made available again. This can be done because a copy of each sample is stored in a priority queue, used to re-insert the sample to the central population; similar to games where characters are re-spawned. In the experiments conducted in this work re-insertion occurs when the population size is below a certain threshold. Figure 1 illustrates the main EvoSpace components.

The population of an EA is stored in-memory using the key-value database Redis, chosen because it provides a hash based implementation of sets and queues which are natural data structures for a PEA model. EvoSpace is implemented as a python module and exposed as a web service using CherryPy. The EvoSpace modules are freely available with a Simplified BSD License at <https://github.com/mariosky/EvoSpace>. The EvoSpace system is deployed using Heroku, a multi-language Platform-as-a-Service

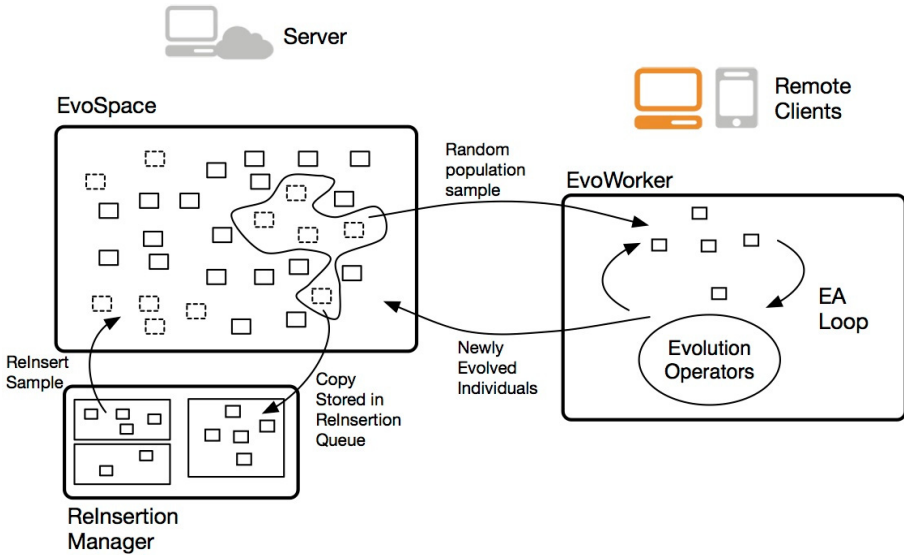


Fig. 1. Main components and dataflow within EvoSpace

(PaaS). The basic unit of composition on Heroku is a lightweight container running a single user-specified process. These containers, which they call *dynos*, can include web (only these can receive http requests) and worker processes (including systems used for database and queuing, for instance). Once deployed the web process can be scaled up by assigning more dynos; in our case and in the more demanding configurations of our experiments, the web process was scaled to 20 dynos. Instructions and code for deployment is available at <https://github.com/mariosky/EvoSpace>. For the experiments carried out for this paper, EvoSpace workers are distributed using the Pi-Cloud PaaS.

4 Problem Statement and Experimental Work

As stated before, one of the main practical issues with EAs is parameter tuning. Following [12], the proposal of the current work is to apply the RPSS approach to a PEA developed with EvoSpace. However, before turning to the experimental work, lets highlight the main differences between a PEA and the Island Model studied in [12, 20]. First, the Island Model is a synchronous EA, while it implements a higher level of parallelization than a normal EA, and is amenable to distributed implementations, it still relies on a synchronized system to perform migration events. Second, the PEA approach based on EvoWorkers can be implemented as a more heterogeneous system than the Island Model, since new EvoWorkers can be added or removed dynamically. In particular, the sample (population) size and number of generations executed by each EvoWorker can be different, since synchronized migrations do not take place. Notice that in EvoSpace, there is no explicit migration process, on the other hand EvoWorkers exchange population members through the centralized pool. These differences could be important

Table 1. Ranges for each EvoWorker parameter

Parameter	Range
Crossover probability	[0,1]
Mutation probability	[0,1]
Sample Size	[12,24]
Generations	[5,30]

regarding the applicability of RPSS on an EvoSpace PEA, which is evaluated in the experimental work.

Therefore, the goal of this paper is to evaluate RPSS on a PEA developed over EvoSpace, in particular a genetic algorithm (GA). In other words, to determine if a random configuration for each of the n EvoWorkers that collaborate on a given run can achieve competitive results. The parameters considered are: 1) crossover probability; 2) mutation probability; 3) sample size; and 4) number of generations (executed locally in each EvoWorker). The valid ranges established for each parameter are summarized in Table 1.

To gauge the effectiveness of RPSS on a PEA, it is compared with three different parametrization strategies, similar to what is done in [12, 20]. All methods are compared based on average performance over a set of runs. First, the simplest approach consists on setting all of the EvoWorker parameters homogeneously. To do this, 200 random parametrizations are created, based on the ranges established in Table 1. The average performance of these runs characterizes the random-homogeneous parametrization, denoted Average-Homogeneous. From these runs, the best configuration is chosen, the one that achieved the best results, and then 20 independent runs are carried out, this method is called Best-Homogeneous¹. Finally, the random-heterogeneous-parametrization is considered, where the parameters of each worker are set independently at random at the beginning of each run; 20 independent runs are performed, the method is denoted as Average-Heterogeneous.

4.1 Benchmark

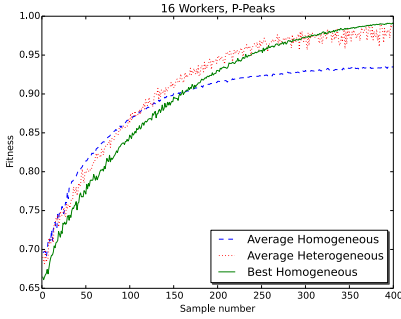
The algorithms are evaluated using the P-Peaks generator of multimodal problems proposed by De Jong et al. [3]. A P-Peaks instance is created by generating a set of P random N -bit strings, which represent the location of the P peaks in the space. To evaluate an arbitrary bit string \mathbf{x} first locate the nearest peak (in Hamming space). Then the fitness of the bit string is the number of bits the string has in common with that nearest peak, divided by N . The optimum fitness for an individual is 1. This particular problem generator is a generalization of the P-peak problems introduced in [4], defined by

$$f_{P-PEAKS}(\mathbf{x}) = \frac{1}{N} \max_{i=1}^P \{N - \text{hamming}(\mathbf{x}, \text{Peak}_i)\}. \quad (1)$$

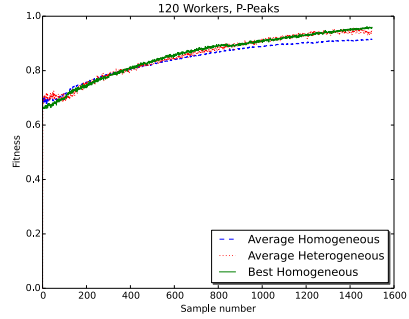
¹ This is a very naive approach to choose the best possible configuration, with much more comprehensive approaches reviewed in [16]. However, here we use the Best-Homogeneous approach for direct comparison with [12, 20].

Table 2. GA configuration for each benchmark problem

Feature	P-Peaks
Crossover (probability)	Two Points (0.7)
Mutation (probability)	Flip Bit (0.2)
Selection	Tournament (size=4)
Variable range	{0, 1}
Survival	Elitist (Keep-Best)
Individuals in the Pool	300,1000 (16,120 workers)



(a) 16 EvoWorkers



(b) 120 EvoWorkers

Fig. 2. Convergence plots for the P-Peaks with 16 (a) and 120 (b) EvoWorkers

A large number of peaks induces a time-consuming search, which is convenient since in order to justify a distributed EA implementation, the cost of computing fitness has to be significantly larger than the implicit communication costs over the network or Cloud. However, according to Kennedy and Spears [14] the length of the string being optimized has a greater effect on the difficulty of the search.

4.2 Experimental Set-up and Results

Experiments are carried out using a different number N of EvoWorkers to solve the benchmark problem. The first group of runs are done with $N = 16$ EvoWorkers, and the second with $N = 120$. Based on [12, 20], it is assumed that with an increased number of workers the RPSS approach should achieve relatively better results, much closer to the Best-Homogeneous configuration. This is particularly important, since increasing the number of EvoWorkers greatly magnifies the dimensionality of the tuning problem. Results are summarized by tracking how the best solution varies with respect to the total number of samples taken from the EvoSpace pool of individuals. These results are presented in Figure 2, where the average performance for each of the three methods evaluated here.

First, for the P-Peaks problem with 16 EvoWorkers we can see a clear trend, the random Heterogeneous configuration is very similar with the best homogeneous configuration, depicted in Figure 2(a). This is a promising initial observation, since the

heterogeneous configuration did not require any parameter tuning, while the best homogeneous configuration is chosen from a set of 200 runs. Moreover, we see that using an homogeneous configuration with random values achieves noticeably inferior performance. When the number of EvoWorkers is increased, shown in Figure 2(b), a similar trend appears, however the differences among the algorithms is reduced. Nevertheless, it is obvious that using a random heterogeneous parametrization can be used as an off-the shelf approach on this problem.

5 Conclusions and Further Work

This paper presents an evaluation of the RPSS parametrization approach on a pool-based EA developed over the EvoSpace system. The basic idea, which is quite simple, is to randomly set the parameter values of each EvoWorker, that connect to the central population pool and perform an independent evolutionary search on a sampled set of individuals. While PEAs developed over EvoSpace have been studied before with good initial results, they suffer from the fact that they have a large number of degrees-of-freedom, requiring extensive parameter tuning. However, using the RPPS approach, it seems that a PEA can be executed successfully without any form of parameter tuning, achieving comparable results to standard homogeneous parametrizations. Future work will focus on exploring the limits of the approach using a more diverse set of benchmark problems, as well as other EA techniques, such as genetic programming or particle swarm optimization.

Acknowledgements. Funding provided by CONACYT (Mexico) Project No. 29537 from the Programa de Estimulo a la Innovación, CONACYT Basic Science Research Project No. 178323, DGEST (Mexico) Research Projects No.5149.13-P and TIJ-ING-2012-110, and IRSES project ACoBSEC from the European Commission. Additional funding provided by projects P08-TIC-03903 (Andalusian Regional Government), TIN2011-28627-C04-02 (Spanish Ministry of Science and Innovation), project 83 (CANUBE) awarded by the CEI-BioTIC UGR. Regional Government Junta de Extremadura, Consejería de Economía, Comercio e Innovación and FEDER, project GRU10029.

References

1. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. John Wiley & Sons (2005)
2. Cantú-Paz, E.: Parameter setting in parallel genetic algorithms. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) *Parameter Setting in Evolutionary Algorithms*. SCI, vol. 54, pp. 259–276. Springer, Heidelberg (2007)
3. De Jong, K.A., Potter, M.A., Spears, W.M.: Using problem generators to explore the effects of epistasis. In: Bäck, T. (ed.) *ICGA*, pp. 338–345. Morgan Kaufmann (1997)
4. De Jong, K.A., Spears, W.M.: An analysis of the interacting roles of population size and crossover in genetic algorithms. In: Schwefel, H.-P., Männer, R. (eds.) *PPSN 1990*. LNCS, vol. 496, pp. 38–47. Springer, Heidelberg (1991)

5. Di Martino, S., Ferrucci, F., Maggio, V., Sarro, F.: Towards migrating genetic algorithms for test data generation to the cloud. In: *Software Testing in the Cloud: Perspectives on an Emerging Discipline*, pp. 113–135. IGI Global (2013)
6. Eiben, A.E., Smith, J.E.: *Introduction to Evolutionary Computing*. Springer (2003)
7. Fazenda, P., McDermott, J., O'Reilly, U.-M.: A library to run evolutionary algorithms in the cloud using mapReduce. In: Di Chio, C., et al. (eds.) *EvoApplications 2012*. LNCS, vol. 7248, pp. 416–425. Springer, Heidelberg (2012)
8. Fernández De Vega, F., Olague, G., Trujillo, L., Lombrana González, D.: Customizable execution environments for evolutionary computation using boinc + virtualization. *Natural Computing* 12(2), 163–177 (2013)
9. García-Valdez, M., Mancilla, A., Trujillo, L., Merelo, J.-J., Fernandez-de Vega, F.: Is there a free lunch for cloud-based evolutionary algorithms? In: *2013 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1255–1262 (2013)
10. García-Valdez, M., Trujillo, L., de Vega, F.F., Merelo Guervós, J.J., Olague, G.: Evospace-interactive: A framework to develop distributed collaborative-interactive evolutionary algorithms for artistic design. In: Machado, P., McDermott, J., Carballeda, A. (eds.) *EvoMUSART 2013*. LNCS, vol. 7834, pp. 121–132. Springer, Heidelberg (2013)
11. García-Valdez, M., Trujillo, L., Fernández de Vega, F., Merelo Guervós, J.J., Olague, G.: EvoSpace: A Distributed Evolutionary Platform Based on the Tuple Space Model. In: Esparcia-Alcázar, A.I. (ed.) *EvoApplications 2013*. LNCS, vol. 7835, pp. 499–508. Springer, Heidelberg (2013)
12. Gong, Y., Fukunaga, A.: Distributed island-model genetic algorithms using heterogeneous parameter settings. In: *IEEE Congress on Evolutionary Computation*, pp. 820–827. IEEE (2011)
13. Holland, J.H.: *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge (1992)
14. Kennedy, J., Spears, W.: Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In: *The 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence 1998*, pp. 78–83 (May 1998)
15. Kramer, O.: *Self-Adaptive Heuristics for Evolutionary Computation*. SCI, vol. 147. Springer, Heidelberg (2008)
16. Lobo, F.G., Lima, C.F., Michalewicz, Z.: *Parameter Setting in Evolutionary Algorithms*. Springer Publishing Company, Incorporated (2007)
17. Merelo-Guervós, J., Castillo, P., Laredo, J.L.J., Mora Garcia, A., Prieto, A.: Asynchronous distributed genetic algorithms with Javascript and JSON. In: *2008 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1372–1379 (June 2008)
18. Sherry, D., Veeramachaneni, K., McDermott, J., O'Reilly, U.-M.: Flex-GP: Genetic programming on the cloud. In: Di Chio, C., et al. (eds.) *EvoApplications 2012*. LNCS, vol. 7248, pp. 477–486. Springer, Heidelberg (2012)
19. Smaoui Feki, M., Nguyen, H.V., Garbey, M.: Parallel genetic algorithm implementation for boinc. In: *PARCO*, pp. 212–219 (2009)
20. Tanabe, R., Fukunaga, A.: Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms. In: *IEEE Congress on Evolutionary Computation*, pp. 1263–1270. IEEE (2013)
21. Trujillo, L., Valdez, M.G., de Vega, F.F., Guervós, J.J.M.: Fireworks: Evolutionary art project based on evospace-interactive. In: *IEEE Congress on Evolutionary Computation*, pp. 2871–2878. IEEE (2013)

PaDe: A Parallel Algorithm Based on the MOEA/D Framework and the Island Model

Andrea Mambrini^{1,*} and Dario Izzo²

¹ University of Birmingham, Birmingham, UK

² European Space Agency, Noordwijk, The Netherlands

Abstract. We study a coarse grained parallelization scheme (thread based) aimed at solving complex multi-objective problems by means of decomposition. Our scheme is loosely based on the MOEA/D framework. The resulting algorithm, called Parallel Decomposition (PaDe), makes use of the asynchronous generalized island model to solve the various decomposed problems. Efficient exchange of chromosomic material among islands happens via a fixed migration topology defined by the proximity of the decomposed problem weights. Each decomposed problem is solved using a generic single objective evolutionary algorithm (in this paper we experiment with self-adaptive differential evolution (jDE)). Comparing our algorithm to MOEA/D-DE we find that it is attractive in terms of performances and, most of all, in terms of computing time. Experiments with increasing numbers of threads show that PaDe scales well, being able to fully exploit the number of underlying available cores.

1 Introduction

In many real-world decision problems several conflicting criteria need to be optimized at the same time. Those problems can be modelled as Multi-objective Optimization Problems (MOPs). A MOP is defined as follow:

$$\begin{array}{ll} \text{Minimize} & F(x) = (f_1(x), \dots, f_o(x)) \\ \text{subject to} & x \in \Omega \end{array}$$

where Ω is the *decision space*, $F(x) : \Omega \rightarrow \mathbb{R}^o$ consists in o *objective functions*, \mathbb{R}^o being the *objective space*. In continuous problems $\Omega \subset \mathbb{R}^s$ and s is defined as the *problem size*.

Being $u = (u_1, \dots, u_o)$ and $v = (v_1, \dots, v_o)$ two objective vectors in Ω , we say that u dominates v if $u_i \leq v_i$ for $i = 1, \dots, o$ and the strict inequality sign holds for at least one objective. A point $x^* \in \Omega$ is called *Pareto Optimal* if there isn't any $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. The set of all pareto optimal points in Ω is called *Pareto Set* and the set of the associated objectives is called *Pareto Front*. The aim of a multi-objective optimisation algorithm is to find a well spread set of points in the Pareto Set, or as close as possible to the Pareto Set.

* Andrea Mambrini has been partially supported by EPSRC through grant no. EP/I010297/1.

Traditionally, multi-objective evolutionary algorithms (MOEAs) are dominance-based. In dominance-based algorithms, individuals are selected using the Pareto dominance notion and some auxiliary criteria aimed at maintaining a good distribution along the same non-dominated front. Examples of dominance-based MOEAs are NSGA-2 [1] and SPEA-2 [2]. Decomposition is another way to solve a MOP. MOEA/D [3] decomposed the original multi-objective problem into many single objective problems constructed in a way that the optimal solution to each of these subproblems is a point on the optimal Pareto Front of the original MOP. It then uses a single-objective optimization algorithm to solve concurrently the subproblems and returns as Pareto Set the union of the optimal solution to each single objective subproblem.

There are two main components of a MOEA/D. The first one is the mechanism to decompose the MOP into subproblems. Traditionally weight vectors are randomly generated and from each of them a single objective problem is obtained using weighted sum [4], Tchebycheff [4] or Boundary Intersection [5]. Some recent work propose a way to adapt the weight vectors throughout the run [6]. The second important component is the way to solve the single objective problems obtained from the decomposition. The original approach uses point crossover and standard mutation [3], while other approaches use different operators, e.g. from Differential Evolutions [7] or Particle Swarm Optimization [8]. As the decomposed problems are solved concurrently, it is natural to consider parallel implementations of a MOEA/D algorithm. In recent work [9] [10] a thread-based parallelization of MOEA/D-DE has been proposed. The approach, a fine-grained scheme, is that of parallelizing the original MOEA/D-DE algorithm by dividing the main population loop in different threads. In order for this fine-grained approach to balance the work load across threads, the number of threads must be kept small in comparison to the population size and the objective function evaluation must account for most of the computing time.

In this paper we introduce a new algorithm called Parallel Decomposition (PaDe) using a different approach to parallelize a MOEA/D algorithm. The idea is to solve each subproblem in a separate logical computational unit and to then use the island model [11] paradigm to introduce exchange of solutions across problems. In an island model, several subpopulations (islands) are evolved independently. Selected individuals are then sent to other islands during a process called *migration*. Island models are well suited for parallelization since group of islands can run in parallel on different computational units. Traditionally in an island model each island runs the same algorithm and solves the same problem. PaDe uses instead a generalised (or heterogeneous) island model where each island solves a different problem and can run a different optimisation algorithm [12]. A recent theoretical work studied how heterogeneous island models can find approximate solutions to NP-Hard problems [13]. Compared to fine-grained parallelization approaches the asynchronous island model at the core of PaDe, is suitable for modern multi-cores architectures as well as for heterogeneous parallel architecture (e.g. grid computing). Moreover the modularity of this approach allow to employ any single objective solver or even run different

ones on each island, thus assigning different areas of the Pareto Front to different algorithms.

We made a C++ implementation of PaDe available as part of the open source scientific library PaGMO [14], and its python front-end PyGMO [15].

Firstly we compare PaDe (with a self-adaptive version of Differential Evolution) to the MOEA/D-DE algorithm [7]. We will show that the two approaches get similar pareto front's quality. We then investigate the parallel performances of PaDe showing how increasing the number of threads up to the population size, PaDe is able, unlike fine grained approach proposed elsewhere [9] [10], to fully take advantage of the underlying cores (linear speedup) even when the objective function has small computational cost.

2 Algorithm Definition

PaDe is a multi-objective evolutionary framework based on decomposition and parallelized using the island model. It first generates m weight vectors of dimension o (being o the number of objectives of the original multi-objective problem) using a weight vector generation method W . The o components of the weight vector must sum to 1, thus the vector must lie on the standard $(o - 1)$ -simplex. The weight vectors can be generated using one of the following methods:

- *GRID*: the weights are generated to optimally maximize their spread as described in [3]. Using this method it is not possible to generate any amount of weight vectors. In fact, for any fixed $H \in \mathbb{N}$ this method can generate $m = \binom{o-1}{H+o-1}$, where o is the dimension of the weight vectors.
- *RANDOM*: differently from GRID it can generate any amount of vectors. It generates each weight vector sampling uniformly at random between 0 and 1 each component. In order to enforce that the sum of all the components of a weight vector is equal to 1, each vector is projected to the o -dimensional standard simplex. This method cannot guarantee the optimality of the spread as the previous method.
- *LOW-DISCREPANCY*: a novel method to generate any amount of weight vectors with a good spread. An Halton sequence [16] of m points of size o is generated. As in the RANDOM method it is then projected to the standard $(o - 1)$ -simplex. This method is a good compromise between the maximum spread guaranteed by the GRID method, and the freedom to generate any amount of vectors guaranteed by RANDOM. We introduced this method as in our island model it is often useful to increase/decrease the population sizes adaptively, which would be not allowed by the standard weight generation method GRID.

After generating the weight vectors, PaDe decomposes the original multi-objective problem into m single objective problems \mathcal{P}_i using one decomposition method between Weighted, Tchebycheff and Boundary Intersection, obtaining each problem from a weight vector w_i . It then assigns each subproblem to an island defined by a single-objective solver S and a population of size $T + 1$. The evolution of the

islands is executed by n threads, each one taking care of evolving m/n islands. In order to fully take advantage of the parallelism, n should be set at least as the number of available computational nodes, so that each core will execute at least one thread. Each island i , associated with the weight vector w_i , is connected to the T islands whose weight vectors are the closest to w_i according to the Euclidian distance: this way migration will exchange solutions between islands solving similar problems.

Then the following is repeated for G_P (PaDe's number of generations) times: each island is evolved using the single objective solver S for G_S generations (solver's number of generations) and, at the end of the evolution, the worst T individuals of each island are replaced from the best individuals from each of the T neighbouring islands (migration). Eventually the union of the best individuals from each island is returned as final population. Algorithm 1 provides an high level pseudo-code description of PaDe.

Algorithm 1. PaDe (Decomposition method D , single-objective solver S , weight generation method W , number of threads n)

Generate m weight vectors w_1, \dots, w_m using W

For each w_i find the set N_i containing the T indices of the Euclidian closest weights

Generate m problems P_i using weights w_i and decomposition method D

Create m random populations \mathcal{P}_i of dimension $T + 1$

Assign each population \mathcal{P}_i to a decomposed problem P_j

for $k = 0$ to G_P **do**

Set $s = 1$

while $s < (m + n)$ **do**

for $i = s$ to $\min(s + n, m)$ **in parallel do**

Migrate solutions from $\mathcal{P}_j, j \in N_i$

Evolve \mathcal{P}_i using S for G_S generations

$s = s + n$

In its current implementation PaDe needs a reference point z^* to be defined upfront and kept fixed throughout the entire run. Such a point could also be defined as the ideal point of the population and updated adaptively during the run, similarly to what done in MOEA/D-DE. In the implementation here discussed this is not done, and z^* is defined as the origin of the axis, which is appropriate as we experiment with DTLZ and ZDT problems. The online update of the z^* point is indeed an area of improvement for PaDE and it is a delicate issue as it must be implemented as to not break the island asynchronicity, while still providing an effective adaptation mechanism for z^* .

2.1 Comparison to a Traditional MOEA/D Implementation

PaDe has several advantages compared to a traditional MOEA/D implementation: the main one being its simple and effective parallelization. In fact, MOEA/D is a steady state algorithm as each individual must be evaluated in sequence. This limit the possibility to solve the subproblems in parallel as it requires synchronization between the nodes and communication at each generation. A simple

transformation of MOEA/D-DE into a generational variant, on the other hand, simply degrades too much performances. In PaDe we use asynchronous migration to help each problem with solutions from the neighbours. This approach is easier to parallelize since each problem is independently solved by every island and communication between islands doesn't need to apply at each generation, while the original performances are kept.

Moreover the island model used by PaDe is asynchronous. That means that each island migrates when it is ready, without waiting for other islands, and each island can receive immigrants at any moment. This and the fact that communication doesn't need to happen at every generation, is particularly helpful when PaDe is deployed on an heterogeneous parallel architecture mixing slow and fast nodes, as for example a grid, where its performances, though, would be affected and need to be evaluated.

The main disadvantage of PaDe is in the overhead caused by the internal island model, and by the population based evolutionary algorithm used on each subproblem, an overhead that is only justified when the decomposed problems are hard, and thus the deployment of generic, state of the art, single objective evolutionary algorithm on the decomposed problems is justified.

Finally PaDe is a flexible and modular algorithm: the decomposition and the optimization of each single-objective subproblem are two phases that can be designed independently. In our implementation [14] we propose a novel method for the former (LOW-DISCREPANCY, see Section 2), while for the latter we provide many well known single objective solvers (Covariance Matrix Adaptation Evolutionary Startegy, Differential Evolution, PSO, Harmony Search, ...).

3 Experiments

The aim of the experiments is to investigate the parallel scalability of PaDe and whether this approach is competitive with other common MOEA/D implementations from a quality of the final Pareto Front point of view. First, we compare PaDe (using a self-adaptive version of Differential Evolution as a solver) with the MOEA/D-DE algorithm [7]. The implementations we used for both the algorithms, are available as part of the open source scientific library PaGMO [14]. We then investigate the parallel performances of PaDe showing how increasing the number of threads up to the population size PaDe is able to fully take advantage of the underlying cores.

3.1 Performance Measures

We have considered the following performance measures

- *p-distance*: measures the average distance between points on a non dominated front and the optimal Pareto Front [17]. The indicator can only be defined for ZDT and DTLZ problems and is zero if the non dominated front belongs to the Pareto Front. A smaller p-distance is better. A zero p-distance means that all points are on the Pareto Front.

- *Hypervolume*: the hypervolume of the Pareto Front calculated according to a reference point shared between both the algorithms and all the runs for the same benchmark problem. The reference point is chosen as the largest point to box all the final non dominated fronts, or equivalently as the nadir point of the union of the populations of all the runs and all the algorithms for the same benchmark problem. Since the reference point is different for each problem we report the hypervolumes normalized as follow $\tilde{h}_i = (h_i - \min(h_1, h_2)) / \max(h_1, h_2)$, where h_1 and h_2 are the hypervolume for PaDe and MOEA/D-DE respectively. When the normalized hypervolume is zero, it indicates that the algorithm achieved the smallest hypervolume. To know how much smaller, one has to read the normalized hypervolume of the other algorithm. Small values will indicate, essentially, that the same quality has been achieved.
- *Fitness Evaluations* this is simply the number of calls to the fitness functions throughout one run. In PaDe the number of fitness evaluations cannot be fixed in advance as the inner algorithm S may have multiple termination conditions (as jDE has). For a fair comparison G_P has been set to get, in almost all cases, a similar number of fitness evaluations between PaDe and MOEA/D-DE.
- *cpu-time* for both the parallel and the sequential experiments this is the wall-clock time for the algorithm to stop after evaluating the given number of generations.

3.2 Sequential Experiments

PaDe running on a single thread has been tested against MOEA/D-DE on the following continuous multi-objective benchmark problems ZDT1, ZDT2, ZDT3, ZDT4, ZDT6 (problem size equal to 30), and for the 3-objectives, 4-objectives and 5-objectives version of DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6, DTLZ7 (problem size equals to 30). For both the algorithms the decomposition method is Tchebycheff, and weight vectors are generated using GRID (see Section 2). The population size has been set to 100 for the 2-objectives problems, 105 for the 3-objectives ones, 120 for the 4-objectives ones and 126 for the 5-objectives one to accommodate the limitation imposed by the grid method (see Section 2). The number of neighbours is set to $T=15$.

PaDe runs for $G_P = 20$ generations on $n = 1$ threads, the solver S used is a self-adaptive differential evolution jDE [18] running for $G_S = 100$ generations. The best/1/exp variant is used. MOEA/D-DE uses a crossover probability equal to 1, it uses both the diversity preserving mechanism described in [7] and it runs for $G_D = 32000$ generation. This has been chosen to have a comparable number of generations between PaDe and MOEA/D. In fact PaDe needs to run the solver on m populations of size $T + 1$. That means that approximately, the number of fitness evaluations required by PaDe is $G_P \cdot m \cdot (T + 1) \cdot G_S$, while MOEA/D-DE will perform approximately $m \cdot G_D$ fitness evaluations. We run the two algorithms 30 times on each problem. We report the median of the measures presented in Section 3.1 on Table 1.

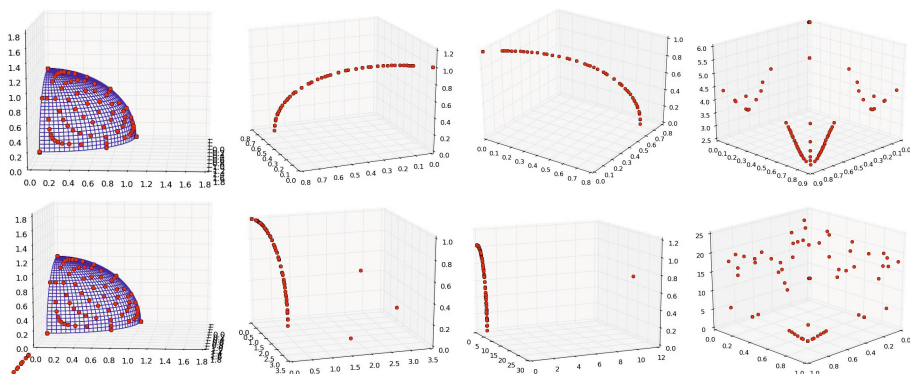


Fig. 1. Comparison between the Pareto Fronts found by MOEA/D-DE (first row) and PaDe (second row) on the 3-objective problems in which PaDe gets a worse p-distance than MOEA/D-DE [DTLZ4,5,6,7 respectively in 1st, 2nd, 3rd and 4th column]

The two algorithms have comparable performances from a quality of the fronts point of view. In Fig. 1 we show the Pareto Fronts for the problems in which PaDe achieves higher p-distance than MOEA/D. Excluding DTLZ7 for which MOEA/D-DE is much better, for the other problems the gap between MOEA/D-DE and PaDe in term of p-distance is due to few point which don't converge to the optimal front rather than to the whole Pareto Front not converging. We suspect this happens because MOEA/D-DE adapts the reference point throughout the run, while PaDe fixes it to the origin. We don't plot the Pareto Fronts for the problems in which PaDe is better because they look very similar to the ones obtained by MOEA/D. From Table 1 we can also notice that the way we set G_S and G_P is appropriate, since it leads to a comparable number of fitness evaluations between MOEA/D and PaDe. Hypervolumes are still very similar between the two methods, while in terms of cpu-time PaDe is faster even if it is executed on a single thread.

3.3 Parallel Experiments

The scalability of PaDe has been tested running it on a 8-core machine with hyperthreading for increasing n (number of threads). Results for the ZDT, DTLZ-3obj, DTLZ-4obj, DTLZ-5obj problems are summarized in Fig. 2. We define the speedup as the ratio between the running time using an increasing number of threads and the running time using just one thread. As all the problems tested do not require to access neither memory nor other peripherals, hyperthreading is not expected to help, thus the maximum theoretical speed-up achievable on the tested architecture is $s_M = 8$.

We see in Fig. 2 how increasing the number of threads up to the population size PaDe is able to fully take advantage of the underlying cores and to get very close to the linear speedup of 8 for most of the problems tested.

Table 1. Sequential experimental results. In grey the best results among the two algorithms. See Section 3.1 for a description of the performance measures.

	p-distance			normalized hypervolume			cpu-time			Fitness evaluations	
	Prob/Alg	PaDe	MOEA/D-DE	PaDe	MOEA/D-DE	PaDe	MOEA/D-DE	PaDe	MOEA/D-DE	PaDe	MOEA/D-DE
ZDT1	1.56e-06	2.04e-05	7.69e-06	0.00e+00	0.00e+00	5.83	12.34	3151320	3200000		
ZDT2	9.77e-07	9.94e-06	0.00e+00	1.77e-05	5.70	12.31	3150480	3200000			
ZDT3	2.26e-01	2.73e-05	0.00e+00	8.65e-05	5.33	13.01	2523240	3200000			
ZDT4	8.89e-09	8.01e-04	2.01e-04	0.00e+00	15.39	21.95	2967360	3200000			
ZDT6	4.15e-01	2.38e-04	0.00e+00	9.89e-05	6.48	13.43	2801200	3200000			
DTLZ-3obj	1.23e-09	3.77e-03	0.00e+00	1.20e-06	20.42	30.45	3050240	3360000			
DTLZ2-3obj	1.38e-11	3.27e-04	0.00e+00	2.02e-05	11.34	20.38	2976920	3360000			
DTLZ3-3obj	5.18e-10	1.42e-03	0.00e+00	2.56e-05	21.08	32.56	2968840	3360000			
DTLZ4-3obj	3.00e-01	8.59e-04	0.00e+00	1.22e-05	12.41	21.95	2952000	3360000			
DTLZ5-3obj	7.16e-02	3.53e-05	0.00e+00	4.31e-05	10.37	20.88	2692640	3360000			
DTLZ6-3obj	5.39e-01	6.93e-06	0.00e+00	4.37e-04	20.46	29.89	3341920	3360000			
DTLZ7-3obj	4.44e-01	2.29e-04	0.00e+00	5.84e-03	6.29	20.43	1774400	3360000			
DTLZ1-4obj	3.68e-10	2.73e-03	0.00e+00	1.89e-05	23.49	40.49	3518080	3840000			
DTLZ2-4obj	3.14e-11	1.43e-04	0.00e+00	9.62e-05	14.60	30.06	3422600	3840000			
DTLZ3-4obj	2.99e-10	8.12e-04	0.00e+00	6.16e-05	26.12	44.63	3411480	3840000			
DTLZ4-4obj	6.37e-01	6.71e-04	0.00e+00	2.13e-05	16.61	31.82	3308120	3840000			
DTLZ5-4obj	1.19e+00	9.07e-01	0.00e+00	1.33e-05	12.82	33.21	2938800	3840000			
DTLZ6-4obj	3.62e+00	3.13e+00	0.00e+00	5.26e-04	25.33	76.12	3811040	3840000			
DTLZ7-4obj	1.42e+00	8.65e-04	0.00e+00	7.74e-03	8.76	28.47	2421680	3840000			
DTLZ1-5obj	2.51e-10	2.20e-03	0.00e+00	2.13e-05	25.06	46.64	3752200	4032000			
DTLZ2-5obj	7.29e-13	4.98e-05	0.00e+00	1.07e-04	17.75	38.20	3691240	4032000			
DTLZ3-5obj	1.90e-12	2.22e-03	0.00e+00	5.76e-05	30.10	53.63	3673960	4032000			
DTLZ4-5obj	1.06e+00	9.13e-05	0.00e+00	1.94e-05	21.25	41.26	3566560	4032000			
DTLZ5-5obj	2.20e+00	1.84e+00	9.66e-04	0.00e+00	14.08	33.54	2802120	4032000			
DTLZ6-5obj	6.70e+00	6.47e+00	0.00e+00	1.06e-05	28.78	75.43	3968080	4032000			
DTLZ7-5obj	1.46e+00	1.42e-03	0.00e+00	5.47e-03	10.04	35.14	2669200	4032000			

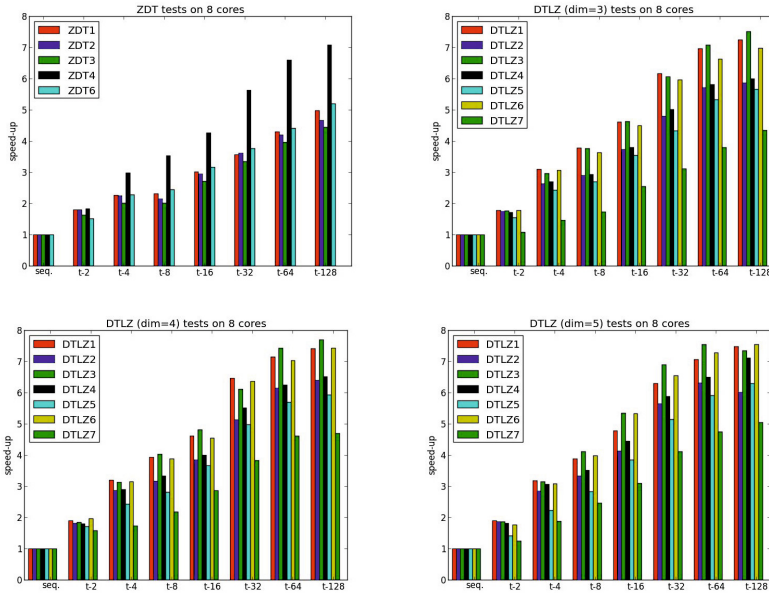


Fig. 2. The number of threads used versus the speedup (ratio between the cpu-time obtained with one thread and the cpu-time obtained with that amount of threads)

4 Conclusions

We have introduced a new multi-objective evolutionary approach, based on the MOEA/D framework and on the island model. The new algorithm, called PaDe is compared to MOEA/D-DE. Experiments show that PaDe can find good approximations to the Pareto Fronts on the tested problems in shorter time than MOEA/D-DE even when deployed on one single CPU. When deployed on multiple CPU architectures, and, unlike fine-grained parallelization approaches for MOEA/D-DE, PaDe is able to provide considerable (close to linear) speed ups also with non CPU intensive fitness landscapes. Moreover the asynchronous island model at the core of PaDe, make the algorithm suitable for modern multi-cores architectures as well as for heterogeneous parallel architecture in which slow nodes are mixed with fast ones (e.g. grid computing).

As future work PaDe should be tested on harder problems, where it could perform better than MOEA/D from a quality of the Pareto Front point of view, and a parallel-safe mechanism to adapt the reference point z^* throughout the run should be implemented to make PaDe competitive also for problems for which setting z^* to the origin is not a sensible choice.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 182–197 (2000)
2. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: *Proceedings of the EUROGEN 2001 Conference* (2001)
3. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evolutionary Computation* 11(6), 712–731 (2007)
4. Miettinen, K.: *Nonlinear Multiobjective Optimization*. International series in operations research & management science. Kluwer Academic Publishers (1999)
5. Das, I., Dennis, J.: Normal-boundary intersection: An alternate method for generating pareto optimal points in multicriteria optimization problems (1996)
6. Jiang, S., Cai, Z., Zhang, J., Ong, Y.S.: Multiobjective optimization by decomposition with pareto-adaptive weight vectors. In: *Seventh International Conference on Natural Computation, ICNC 2011* (2011)
7. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation* 13(2), 284–302 (2009)
8. Al Moubayed, N., Petrovski, A., McCall, J.: A novel smart multi-objective particle swarm optimisation using decomposition. In: Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G. (eds.) *PPSN XI. LNCS*, vol. 6239, pp. 1–10. Springer, Heidelberg (2010)
9. Nebro, A.J., Durillo, J.J.: A Study of the Parallelization of the Multi-Objective Metaheuristic MOEA/D. In: Blum, C., Battiti, R. (eds.) *LION 4. LNCS*, vol. 6073, pp. 303–317. Springer, Heidelberg (2010)
10. Durillo, J.J., Zhang, Q., Nebro, A.J., Alba, E.: Distribution of Computational Effort in Parallel MOEA/D. In: Coello, C.A.C. (ed.) *LION 2011. LNCS*, vol. 6683, pp. 488–502. Springer, Heidelberg (2011)
11. Tomassini, M.: *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time*. Springer (2005)
12. Izzo, D., Ruciński, M., Biscani, F.: The generalized island model. In: Fernandez de Vega, F., Hidalgo Pérez, J.I., Lanchares, J. (eds.) *Parallel Architectures & Bioinspired Algorithms. SCI*, vol. 415, pp. 151–170. Springer, Heidelberg (2012)
13. Mambrini, A., Sudholt, D., Yao, X.: Homogeneous and heterogeneous island models for the set cover problem. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I. LNCS*, vol. 7491, pp. 11–20. Springer, Heidelberg (2012)
14. PaGMO: Parallel Global Multiobjective Optimizer,
<http://pagmo.sourceforge.net/pagmo/>
15. PyGMO: Python Parallel Global Multiobjective Optimizer,
<http://pagmo.sourceforge.net/pygmo/>
16. Halton, J.H.: Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM* 7(12), 701–702 (1964)
17. Märten, M., Izzo, D.: The asynchronous island model and NSGA-II: study of a new migration operator and its performance. In: *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, pp. 1173–1180. ACM (2013)
18. Brest, J., Zumer, V., Maucec, M.S.: Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: *IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 215–222. IEEE (2006)

Evolution-In-Materio: Solving Machine Learning Classification Problems Using Materials

Maktuba Mohid¹, Julian Francis Miller¹, Simon L. Harding², Gunnar Tufte²,
Odd Rune Lykkebo³, Mark K. Massey³, and Michael C. Petty

¹ Department of Electronics, University of York, York, UK
{mm1159,julian.miller}@york.ac.uk, slh@evolutioninmaterio.com

² Department of Computer and Information Science,
Norwegian University of Science and Technology, 7491 Trondheim, Norway
{gunnart,lykkebo}@idi.ntnu.no

³ School of Engineering and Computing Sciences and Centre for Molecular and
Nanoscale Electronics, Durham University, UK
{m.k.massey,m.c.petty}@durham.ac.uk

Abstract. Evolution-in-materio (EIM) is a method that uses artificial evolution to exploit the properties of physical matter to solve computational problems without requiring a detailed understanding of such properties. EIM has so far been applied to very few computational problems. We show that using a purpose-built hardware platform called Mecobo, it is possible to evolve voltages and signals applied to physical materials to solve machine learning classification problems. This is the first time that EIM has been applied to such problems. We evaluate the approach on two standard datasets: Lenses and Iris. Comparing our technique with a well-known software-based evolutionary method indicates that EIM performs reasonably well. We suggest that EIM offers a promising new direction for evolutionary computation.

Keywords: Evolutionary algorithm, evolution-in-materio, material computation, evolvable hardware, machine learning, classification problem.

1 Introduction

Natural evolution could be viewed as an algorithm which exploits the physical properties of materials. Evolution-in-materio (EIM) aims to mimic the exploitation of physical properties by natural evolution by manipulating physical systems using computer controlled evolution (CCE) [6,10]. In particular, EIM aims to exploit the properties of physical systems for solving computational problems.

Evolution-in-materio was first described by Miller and Downing [10]. The concept was inspired by the work of Adrian Thompson who investigated whether it was possible to evolve working electronic circuits using a silicon chip called a Field Programmable Gate Array (FPGA). He evolved a digital circuit that could discriminate between 1kHz or 10kHz signal [12]. When the evolved circuit was analysed Thompson discovered that artificial evolution had exploited physical

properties of the chip. To demonstrate that EIM was possible, Harding and Miller attempted to replicate these findings using a liquid crystal display. They found that computer-controlled evolution could utilize the physical properties of liquid crystal to help solve a number of computational problems [4]:

- Two input logic gates: OR, AND, NOR, NAND, etc. [6].
- Tone Discriminator: A device was evolved which could differentiate different frequencies [4].
- Robot Controller: A controller for a simulated robot with wall avoidance behavior [5].

In this paper, we describe the use of a purpose built platform called Mecobo that facilitates computer controlled evolution of a material (the hardware is described in detail in [8]). The Mecobo platform has been developed within an EU funded research project called NASCENCE [3]. The computational material we have used in this investigation is a mixture of single-walled carbon nanotubes and a polymer. Evolutionary computation has been widely used to solve machine learning classification problems. Here, we show that using the Mecobo platform it is possible to evolve solutions to two classification problems using materials. To form a basic assessment of effectiveness of the technique we have compared our results with a well-known software-based evolutionary computation technique called Cartesian Genetic Programming (CGP) [9] on the same problems.

The organisation of the paper is as follows. In Sect. 2 we give a brief conceptual overview of EIM. We describe the Mecobo EIM hardware platform in Sect. 3. The preparation and composition of the physical computational material is described in Sect. 4. Sect. 5 describes the machine learning classification problem. The way we have used the Mecobo platform for classification problem is described in Sect. 6. We describe our experiments and analysis of results in Sect. 7. Finally we conclude and offer suggestions for further investigation in Sect. 8.

2 Conceptual Overview Of Evolution-In-Materio

EIM is a hybrid system involving both a physical material and a digital computer. In the physical domain there is a material to which physical signals can be applied or measured. These signals are either input signals, output signals or configuration instructions. A computer controls the application of physical inputs applied to the material, the reading of physical signals from the material and the application to the material of other physical inputs known as physical configurations. A genotype of numerical data is held on the computer and is transformed into configuration instructions. The genotypes are subject to an evolutionary algorithm. Physical output signals are read from the material and converted to output data in the computer. A fitness value is obtained from the output data and supplied as a fitness of a genotype to the evolutionary algorithm [11]. Figure 1 shows conceptual overview of EIM.

Miller and Downing noted that only certain materials may be suitable for EIM and they gave some guidelines for choosing materials [10]. The material needs to

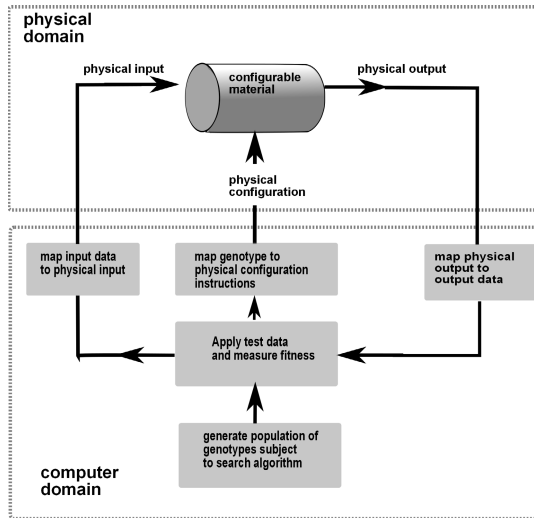


Fig. 1. Concept of evolution-in-materio [11]

be reconfigurable, i.e., it should be able to be evolved over many configurations to get the desired response. It is also important for a physical material to be able to be “reset” in some way before applying new input signals to it, otherwise it might preserve some memory of a past configuration and give fitness scores that are dependent on the past. Preferably, the material should also be able to be physically configured using small voltage and be manipulable at a molecular level.

3 Mecobo Hardware Platform

The hardware system we have used has three main components: a host computer, the Mecobo platform and an electrode array. The Mecobo platform is designed to interface a large variety of materials. The hardware allows the possibility to map inputs, outputs, configurations and signal properties in arbitrary ways. The platform’s software components, i.e. the EA and the software stack, are as important as the hardware. Mecobo includes a flexible software platform including hardware drivers, support of multiple programming languages and the possibility to connect to hardware over the internet. This makes Mecobo a highly flexible platform for EIM experimentation [8].

Mecobo is built on PCBs with an FPGA as the main component. The digital and analogue parts of Mecobo are implemented on separate PCBs. All the analogue components are placed on a daughter board; such as crossbar switches and analogue-digital converters. This has the advantage that it allows the redesign of the analogue part of the system without changing the digital part of the motherboard. A micro controller stands as a communication interface between the FPGA and an external USB port.

At present the Mecobo hardware allows only two types of inputs to the material. One is constant voltage (0V or 3.5V) and the other is a square wave signal.

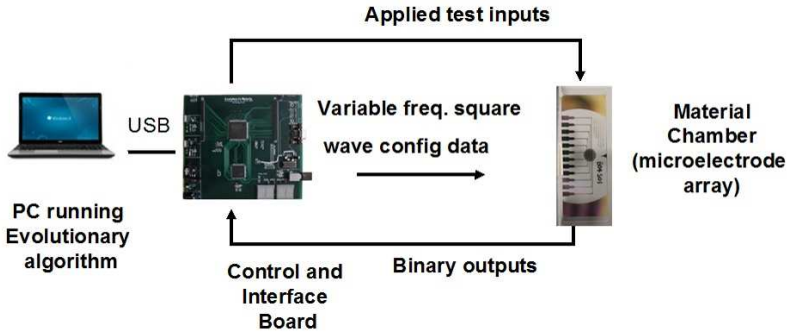


Fig. 2. Mecobo Hardware platform together with gold electrode array and material sample

Different characteristics or input parameters associated with these inputs can be chosen, these control the amplitude of an input, the square wave frequency, cycle time (percentage of period square wave is 1), phase, and the start and stop times of applied input. The start time and end time of each input signal determine how long an input is applied. Mecobo only samples using digital voltage thresholds, hence the material output is interpreted as strictly high or low, (i.e. 0 or 1). In later versions of this hardware, analogue inputs and outputs will be possible.

In the case that an electrode is chosen to be read (see section 4), a user-defined output sampling frequency determines the buffer size of output samples. If the output frequency is F_{out} , start time $Time_{start}$ and end time is $Time_{end}$, then the buffer size is Buf_{size} is given by:

$$Buf_{size} = F_{out}(Time_{end} - Time_{start})/1000 \quad (1)$$

Where, $Time_{start}$ and $Time_{end}$ are measured in milliseconds. However, in practice due to pin latency, the real buffer size is generally smaller.

4 Physical Computational Material

The experimental material consists of single-walled carbon nanotubes mixed with polymethyl methacrylate (PMMA) and dissolved in anisole (methoxybenzene) ¹. The sample is baked causing the anisole to evaporate. This results in material which is mixture of carbon nanotube and PMMA. The concentration of carbon nanotube is 0.71% (weight% fraction of PMMA). Carbon nanotubes are conducting or semi-conducting and role of the PMMA is to introduce insulating regions within the nanotube network, to create non-linear current versus voltage characteristics. Another benefit of the polymer is to help with dispersion of the nanotubes in solution. The preparation of experimental material is given below:

¹ Mark K. Massey and Michael C. Petty prepared the materials used as substrates and the electrode masks for our experiments.

- A M3 sized nylon washer was glued on the electrode array to contain the material whilst drying;
- 20 μL of material were dispensed into the washer;
- This was dried at $\approx 100^\circ \text{C}$ for $\approx 1 \text{ h}$ to leave a “thick film”.

The experimental material is placed in the middle of a plate of the electrode array. Twelve gold electrodes are connected directly with the experimental material in the plate. The electrode array is connected directly with the Mecobo board via wires. The electrode sample is shown in Fig. 2.

5 Machine Learning: Classification Problems

Classification is an important class of problems in Machine learning. The objective is to correctly classify data instances. In this paper we have evaluated our approach on two classification problems: Lenses and Iris. [2]. Both datasets have four attributes which are classified into one of three classes. The Lenses dataset consists of 24 instances with integer attributes. The attributes are categorical in nature and take values either 1,2 or 1, 2, 3. We used the first 16 instances as training data and the last 8 as testing data. The Iris dataset contains 150 instances with real-valued attributes. The first fifty instances are class 1, the second fifty class two and third set of 50 are class 3. We divided the data set into two groups (training and testing set) of 75 instances each. Each set contained exactly 25 instances of each class.

6 Classifying Data Using Evolution-In-Materio

6.1 Methodology

The experiments were performed with an electrode array having 12 electrodes. For both datasets, four electrodes have been used as inputs (i.e. they are attribute related), 3 electrodes have been used as outputs (i.e. to define the class) and the remaining 5 electrodes have been used for configuration voltages. Each output electrode is associated with an output class. Each chromosome defined which electrodes are either outputs, inputs (receive square waves) or receive the configuration data (square waves or constant voltage). We accumulated sampled output values in a buffer for 128 milliseconds using a 25KHz sampling frequency.

The fitness calculation in the evolutionary algorithm only used training data. Once the evolutionary algorithm finished the configuration of electrodes having the best fitness was subsequently tested with the test data to determine its ability to predict correctly unseen data (the test set).

6.2 Genotype Representation

Each chromosome used $n_e = 12$ electrodes at a time. Associated with each electrode there were six genes which define which electrode was used, or characteristics of the input applied to the electrode: signal type, amplitude, frequency,

Table 1. Description of genotype

Gene Symbol	Signal applied to, or read from i^{th} electrode	Allowed values
p_i	Which electrode is used	0, 1, 2 ... 11
s_i	Type	0 (constant), 1 (square-wave)
a_i	Amplitude	0, 1
f_i	Frequency	500, 501 ... 10K
ph_i	Phase	1, 2 ... 10
c_i	Cycle	0, 1, 2 ... 100

phase, cycle (see Sect. 3). This means that each chromosome required a total of 72 genes. Mutational offspring were created from a parent genotype by mutating a single gene (i.e., one gene of 72). The values that genes could take are shown in Table 1, where i takes values 0, 1, ... 11.

The genotype for a chromosome of an individual consists of the 72 genes shown below:

$$p_0s_0a_0f_0ph_0c_0 \dots p_{11}s_{11}a_{11}f_{11}ph_{11}c_{11}$$

6.3 Input Mapping

The inputs to the electrode array (representing the data instances) were square waves of a particular frequency. The frequency was determined by a linear mapping of attribute data. Denote the i^{th} attribute in a dataset by I_i , where i takes values 1, 2, 3, 4. Denote the maximum and minimum value taken by this attribute in the whole data set by $I_{i_{max}}$ and $I_{i_{min}}$ respectively. Denote the maximum and minimum allowed frequencies be denoted by F_{max} and F_{min} respectively. Then the linear mapping given in Eqn. 2 allows the i^{th} attribute of an instance I_i to map to a square-wave frequency F_i which was applied to a given electrode. In the experiments we chose $F_{min} = 500\text{Hz}$ and $F_{max} = 10000\text{ Hz}$.

$$F_i = a_i I_i + b_i \quad (2)$$

where the constants a_i and b_i are found by setting I_i and F_i to their respective maximum and minimum and solving for a_i and b_i .

$$a_i = (F_{max} - F_{min}) / (I_{i_{max}} - I_{i_{min}}) \quad (3)$$

$$b_i = (F_{min} I_{i_{max}} - F_{max} I_{i_{min}}) / (I_{i_{max}} - I_{i_{min}}) \quad (4)$$

6.4 Output Mapping

We determined the class that an instance belonged to, by examining the output buffers which contain samples taken from the output electrodes. The current Mecobo platform can only recognize binary values, so the output buffers contain a binary string. We used the *transitions* from 0 to 1 in the output buffers to define the class that an instance belonged to. For each output buffer, the positions of transitions were recorded and the gaps between consecutive transitions were

measured and an average calculated. A transition based fitness was used as it is frequency related. Since instance data determine the frequencies of applied signals, it seemed natural to use a method of reading output buffer bitstrings that is itself frequency related. An example of average gap calculation for an output electrode has been shown in Fig. 3.

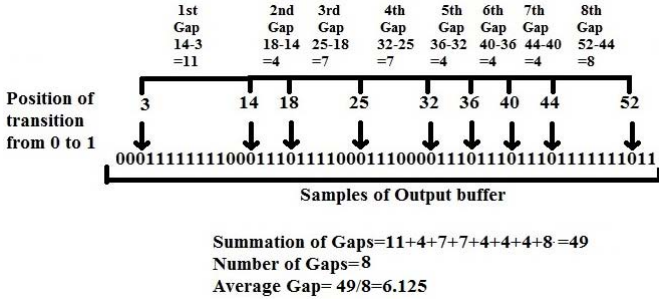


Fig. 3. Example of average transition gap calculation for an output electrode

The output class was determined by the output buffer with largest average transition gap. If two or more buffers had the same average gap then the class was determined by the first such buffer encountered (starting at 1).

6.5 Fitness Score

The fitness calculation required counts to be made of the number of true positives TP , true negatives TN , false positives, FP and false negatives, FN . For an instance having a class c , according to the dataset, and a predicted class p , we can calculate TP , TN , FP , and FN as shown in Eqn. 5.

$$\begin{aligned}
 &\text{if } p = c \text{ then } TP = TP + 1; TN = TN + 2 \\
 &\text{if } p \neq c \text{ then } FP = FP + 1; FN = FN + 1; TN = TN + 1
 \end{aligned}
 \tag{5}$$

The explanation of this is as follows. If the predicted p is correct, then it is a true positive so TP should be incremented. It is also a true negative for the other two classes, hence TN should be increased by two. If the predicted class is incorrect, then it is a false positive for the class predicted, so FP should be incremented. It is also a false negative for the actual class of the instance, so FN should be incremented. Finally, the remaining class is a true negative, so TN should be incremented. Once all instances have been classified we calculated the fitness of a genotype using Eqn. 6 [1].

$$fitness = \frac{TP.TN}{(TP + FP)(TN + FN)}
 \tag{6}$$

So if all instances are correctly predicted, the fitness is 1, since in this case $FP = 0$ and $FN = 0$. In the case that all instances are incorrectly predicted, then $TP = 0$ and $TN = 0$, so the fitness is zero.

7 Experiments

For each of the datasets a $1 + \lambda - ES$ evolutionary algorithm with $\lambda = 4$ was used [9] and run for 500 generations. This evolutionary algorithm has a population size of $1 + \lambda$ and selects the genotype with the best fitness to be the parent of the new population. If there is no offspring better than the parent but at least one with a fitness equal to the parent, then an offspring is chosen to be the new parent. The remaining members of the population are formed by mutating the parent. Thirty and twenty independent runs were carried out for the Lenses dataset and Iris dataset respectively. The smaller number of runs for the latter was due to the large time required for each experiment. It took more than 12 hours to run 500 generations on the Iris training set. This time also precluded using leave-one-out cross validation methods.

7.1 Using CGP For Classification

To evaluate the effectiveness of the EIM method for solving classification problems we compared results with those obtainable using CGP using the same $1 + 4$ evolutionary algorithm over the same number of generations using the same fitness function. It should be noted that CGP has previously been shown to perform well on classification problems (e.g. with Mars terrain images [7] and mammograms [13]). CGP is a graph-based form of genetic programming [9]. The genotypes encode directed acyclic graphs and the genes are integers that represent where nodes get their data, what operations nodes perform on the data, and where the output data required by the user is to be obtained. In classification problems the number of outputs, n_O is chosen to be equal to the number of classes in the dataset. The class of a data instance is defined as the class indicated by the maximum numerical output. The function set chosen for this study was defined over the real-valued interval $[0.0, 1.0]$ and consisted of the following primitive functions of their inputs. The functions were assumed to have three inputs, z_0, z_1, z_2 (but some are ignored):

$(z_0 + z_1)/2$; $(z_0 - z_1)/2$; $z_0 z_1$;
if $|z_1| < 10^{-10}$ **then** 1 **else if** $|z_1| > |z_0|$ **then** z_0/z_1 **else** z_1/z_0 ;
if $z_0 > z_1$ **then** $z_2/2$ **else** $1 - z_2/2$.

We used *three* mutation parameters. A percentage for mutating connections, μ_c and functions, μ_f . Mutation of outputs μ_o , is done probabilistically. In all experiments $\mu_c = 3\%$, $\mu_f = 3\%$, and $\mu_o = 0.5$. The output mutation probability was set as 0.5 because there are only as many outputs as there are classes. We chose a linear CGP geometry by setting the number of rows, $n_r = 1$ and the number of columns, $n_c = 100$ with nodes being allowed to connect to any previous node.

Table 2. Experimental results comparing results of experimental material with CGP on machine learning classification problem using two datasets: Lenses and Iris. Accuracy is the percentage of the training or test set correctly predicted.

Dataset	Average Training Accuracy of Experimental Material	Average Test Accuracy of Experimental Material	Best Accuracy of Experimental Material	Average Training Accuracy of CGP	Average Test Accuracy of CGP	Best Accuracy of CGP
Lenses	92.7%	65.8%	95.8%	93.8%	68.3%	95.8%
Iris	84.7%	77.1%	96.7%	97.7%	93.6%	98.0%

7.2 Results and Discussion

It can be seen from Table 2 that in the case of the Lenses dataset the training and testing of experimental material are very close to the corresponding accuracies of CGP and best accuracy of experimental material is same as that of CGP. In the case of the Iris dataset, although the results with training and test for the experimental material are not as good as CGP, the best accuracy is quite close.

8 Conclusions and Future Outlook

We have shown how using a purpose-built evolutionary platform called Mecobo, we can evolve configurations of a physical system to perform classification. The material we have used is a mixture of single-walled carbon nanotubes and a polymer. The aim of the paper is not to show that the experimental results of solving machine learning classification problems using EIM is competitive with state-of-the-art machine learning classification algorithms, but rather to start a new beginning in the world of computation. To our knowledge, this is the first time that classification problems have been attempted by the manipulation of a physical material. There were many decisions that were made in this investigation that require more detailed experiments before the ideal experimental conditions can be ascertained. This implies that it is likely that much better results could be obtained in the future. Increasing the number of electrodes could allow us to consider more instances or instances with more attributes, this could make the system faster and scale up to larger problems. Circuitry could be potentially built that allows the electrode array and material sample to act as a standalone classifier (i.e. no PC, or Mecobo board). There remain many questions for the future. The Mecobo platform is currently under development and the next version will be able to allow the utilization of analogue voltages.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 317662.

References

1. Akbarzadeh, V., Sadeghian, A., dos Santos, M.: Derivation of relational fuzzy classification rules using evolutionary computation. In: IEEE Int. Conf. on Fuzzy Systems, pp. 1689–1693 (2008)
2. Bache, K., Lichman, M.: UCI machine learning repository (2013), <http://archive.ics.uci.edu/ml>
3. Broersma, H., Gomez, F., Miller, J.F., Petty, M., Tufte, G.: Nascence project: Nanoscale engineering for novel computation using evolution. *International Journal of Unconventional Computing* 8(4), 313–317 (2012)
4. Harding, S., Miller, J.F.: Evolution in materio: A tone discriminator in liquid crystal. In: Proc. Congress on Evolutionary Computation 2004, vol. 2, pp. 1800–1807 (2004)
5. Harding, S., Miller, J.F.: Evolution in materio: A real time robot controller in liquid crystal. In: Proc. NASA/DoD Conference on Evolvable Hardware, pp. 229–238 (2005)
6. Harding, S.L., Miller, J.F.: Evolution in materio: Evolving logic gates in liquid crystal. *Int. J. of Unconventional Computing* 3(4), 243–257 (2007)
7. Leitner, J., Harding, S., Forster, A., Schmidhuber, J.: Mars terrain image classification using cartesian genetic programming. In: 11th International Symposium on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS (2012)
8. Lykkebø, O.R., Harding, S., Tufte, G., Miller, J.F.: Mecobo: A Hardware and Software Platform for In Materio Evolution. In: Ibarra, O.H., Kari, L., Kopecki, S. (eds.) UCNC 2014. LNCS, vol. 8553, pp. 267–279. Springer, Heidelberg (2014), http://dx.doi.org/10.1007/978-3-319-08123-6_22
9. Miller, J.F. (ed.): Cartesian Genetic Programming. Springer (2011)
10. Miller, J.F., Downing, K.: Evolution in materio: Looking beyond the silicon box. In: Stoica, A., Lohn, J., Katz, R., Keymeulen, D., Zebulum, R.S. (eds.) The 2002 NASA/DoD Conference on Evolvable Hardware, vol. 7, pp. 167–176. IEEE Computer Society (2002)
11. Miller, J.F., Harding, S.L., Tufte, G.: Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence* 7, 49–67 (2014)
12. Thompson, A.: Hardware Evolution - Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution. Springer (1998)
13. Völk, K., Miller, J.F., Smith, S.L.: Multiple network CGP for the classification of mammograms. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 405–413. Springer, Heidelberg (2009)

An Analysis of Migration Strategies in Island-Based Multimemetic Algorithms

Rafael Nogueras and Carlos Cotta

Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga,
ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain
ccottap@lcc.uma.es

Abstract. Multimemetic algorithms (MMAs) are memetic algorithms that explicitly represent and evolve memes (computational representations of problem solving methods) as a part of solutions. We consider an island-based model of MMAs and provide a comparative analysis of six migrant selection strategies and two migrant replacement operators. We use a test suite of four hard pseudoboolean functions to examine qualitative behavioral differences at the genetic and memetic level, and provide a sound statistical analysis of performance. The results indicate the choice of migrant selection operator is more important than that of migrant replacement, and that policies based on fitness or pure genetic diversity do not compare favorably to more holistic strategies.

1 Introduction

Memetic optimization [11] is a long standing search paradigm conceived as a pragmatic combination of population-based global search techniques and trajectory-based local search techniques. The notion of *meme* as unit of imitation (ultimately translating to local-improvement procedures in this computational context) is central to this paradigm. While many simple memetic approaches rely on predefined local-search procedures (i.e., static memes), the idea of explicitly exploiting the computational evolution of memes has been around for some time [10] and is now the central tenet of memetic computing [13] defined as “...a paradigm that uses the notion of meme(s) as units of information encoded in computational representations for the purpose of problem solving”. Such an explicit treatment of memes can be found in multimemetic algorithms (MMAs) [9], in which solutions carry memes indicating how they are going to self-improve.

An important issue in such MMAs is the way in which memes propagate throughout the population. In this sense, meme propagation dynamics is more complex than that of their genetic counterparts, if only because memes are only indirectly evaluated according to the effect they exert on the solutions they are attached to (hence, mismatches between genes and memes may cause potentially good memes become extinct or poor memes proliferate [12]). These issues are specifically relevant to multi-population models of MMAs, in which in addition to internal population dynamics one also has to consider the effect of the communication among populations. Although the influence of the migration policy

has been well-studied in the context of evolutionary algorithms –e.g., [1, 3, 14]; see also [5, 17]– to the best of our knowledge it has not been attempted on this family of MMAs. Notice that in addition to the role that the migration policy can have on properties such as population diversity, in this family of techniques individuals are also responsible for conducting actively part of the search in a self-adaptive way, and carry information for this purpose. Hence design decisions regarding migration do not just affect implicitly the search process via gene diffusion but do it explicitly by means of meme propagation. In this work we take some steps in this direction and provide a comparative analysis of migration policies on a class of MMAs.

2 Island-Based Multimemetic Model

To analyze the impact that the choice of migration strategies has on island-based MMAs, let us firstly describe the basic underlying model and then go to detail the migration policies considered in the experimentation.

2.1 Basic Algorithmic Model

Our MMA is close in spirit to the model defined by Smith [15] in which each individual in the population carries a binary genotype and a single meme. The latter represents a rewriting rule expressed as a pair $\langle \textit{condition}, \textit{action} \rangle$ as follows: let $\langle C, A \rangle$ be a rule, with $C, A \in \Sigma^r$ where $\Sigma = \{0, 1, \#\}$ is a ternary alphabet in which ‘#’ represent a wildcard symbol; now, given a genotype $b_1 b_2 \cdots b_n$, a rule $\langle c_1 \cdots c_r, a_1 \cdots a_r \rangle$ could be potentially applied on any part of the genotype into which the condition fits, i.e., $b_i b_{i+1} \cdots b_{i+r-1} = c_1 \cdots c_r$ (wildcard symbols in the right hand side are assumed to match any symbol in the left hand side). Were the rule applied on a site i , its action would be to implant the action $A = a_1 \cdots a_r$ in that part of the genotype, i.e., letting $b_i b_{i+1} \cdots b_{i+r-1} \leftarrow a_1 \cdots a_r$ (here, wildcard symbols in the right hand side are interpreted as don’t-change symbols, leaving the corresponding symbol in the left hand side unchanged). To avoid positional bias, the order in which the genotype is scanned is randomized. Once a match is found the rule is applied and the resulting neighboring genotype is evaluated. In order to keep the total cost of the process under control, a parameter w which determines the maximal number of rule applications per individual is used. The best neighbor generated (if better than the current genotype) is kept.

Besides the use of memes embedded within individuals, our MMA otherwise resembles a standard memetic algorithm in which parents are selected using binary tournament, and in which recombination, mutation and local-search (conducted using the meme linked to the individual) are used to generate the offspring, which replaces the worst parent following the model presented in [12].

2.2 Migration Strategies Considered

In order to deploy the MMA described before on a multi-island model it is necessary to define a interconnection topology (e.g., a ring, a grid, a hypercube, etc.)

and a migration policy. Such a policy encompasses determining parameters such as the number m of individuals undergoing migration, the frequency ζ of such events, the procedure ω_S used to select the individuals to be migrated from the emitting island, the procedure ω_R used to handle migrants in the receiving island, and the synchronous/asynchronous character of the interaction – see [1]. In this work we are going to consider synchronous interaction and we will be specifically concerned about ω_S and ω_R , whose nature is qualitative as opposed to the quantitative nature of the numerical parameters m and ζ , and whose study cannot therefore be approached using a numerical tuning approach.

Regarding the migrant selection operator ω_S , we have considered the following six possibilities:

- **best**: the best m individuals in the emitting population are selected for migration. This strategy could be seen as an attempt to provide the maximal immediate boost in fitness in the receiving island, probably inducing the latter to re-focus the search if the migrants start to takeover the population.
- **random**: migrants are selected by random sampling (without replacement) of the emitting population. In this case, the goal is injecting diversity in the target population by providing a random sample of the genetic/memetic material of the emitting island.
- **probabilistic**: this strategy borrows inspiration from estimation of distribution algorithms and is related in spirit to the previous strategy. Here, a probabilistic model of the emitting population is created and used to produce the migrants. Hence, these provide a sample of the information contained in the emitting island but do not necessarily correspond to existing individuals in the latter. It can thus be seen as more exploratory than random selection. In this work we consider a simple univariate model in which migrants are generated so that the probability of each symbol in a given position matches the relative frequency of that symbol in that position in the population.
- **diverse-gene**: in the line of the multikulty algorithm [2], migrants are here selected so as to introduce as much diversity as possible in the target population, cf. [4]. To this end, individuals whose genotypic distance (in a Hamming sense) to individuals in the receiving population is maximal are selected.
- **diverse-meme**: this is the natural extension of the previous strategy to the memetic realm. In this case, migrants are individuals carrying memes whose distance (again in a Hamming sense) to memes in the receiving island is maximal. The goal is thus not introducing explicit genetic diversity but do this implicitly by introducing diversity in the way solutions are improved.
- **random-immigrants**: this strategy generates the migrants completely at random whenever they are needed [7]. Since it does not take into account the emitting island at all, this strategy represents an attempt to measure the raw effect of introducing new individuals in the target population, decoupling it from the effects attributed to the actual information exchange between islands. In some sense, it thus provides a performance baseline above which the performance of the other strategies could be assessed.

As to the migrant replacement operator ω_R , we have considered the following two possibilities:

- **replace-worst**: the worst individuals in the population are replaced by the incoming migrants.
- **replace-random**: the migrants replace randomly selected individuals.

In either case we choose to perform the replacement unconditionally (i.e., the migrants are always accepted in the target population) for two reasons: firstly, we aim to maximize the effects (positive or negative) of the migration operation, and secondly we promote diversity over immediate fitness loss (recall that in MMAs, solutions are subject to local improvement and hence such losses can be relieved via meme application; furthermore, exploring the basins of attraction of other optima may be a more valuable asset than a good quality solution in a well-represented –by other solutions in the population– basin of attraction).

3 Experimental Analysis

The migration strategies introduced in the previous section have been subject to experimental scrutiny. Before presenting the actual results, next section describes the experimental setting and test suite considered in the experimentation.

3.1 Benchmark and Settings

The MMA has been tested using the following four different problems defined on binary strings:

- Deb’s 4-bit fully deceptive function (TRAP henceforth) [6]. In our experiments we have considered the concatenation of $k = 32$ 4-bit traps (i.e., 128-bit strings, $opt = 32$).
- Watson et al.’s hierarchically consistent test problems (HIFF and HXOR) [16]. These are recursive epistatic problems defined on 2^k -bit strings which force the algorithm to search for combinations of increasingly larger building blocks. We have considered $k = 7$ (i.e., 128-bit strings, $opt = 576$).
- Boolean satisfiability: a classical NP-complete problem in which a truth assignment to n variables has to be found in order to satisfy a certain Boolean formula Φ . We consider this formula is expressed in conjunctive normal form with $n = 128$ variables and $k = 3$ variables per clause. We use a problem generator approach, generating a different satisfiable instance with the critical clauses/variable ratio ($opt = m = 4.3n = 550$) in each run of the MMA.

We consider an island-based MMA (iMMA) as described in Sect. 2.1, with a population size of $\mu = 128$ individuals, recombination probability $p_X = 1.0$ and mutation probability $p_M = 1/\ell$ ($\ell = 128$, the genome length). This population is arranged in $n_i \in 1, 2, 4, 8$ islands, each of them comprising μ/n_i individuals. The case $n_i = 1$ (denoted as sMMA) corresponds to panmixia and involves no migration whatsoever. In the remaining scenarios, the islands are arranged

in a unidirectional ring and migration takes place every $\zeta = 20$ generations, thus allowing a reasonable lapse of isolated evolution in each deme, cf. [1]. One migrant is selected using ω_S and inserted in the receiving population using ω_R , where both ω_S, ω_R are the strategies described in Sect. 2.2. The memes are expressed as rules of length $r = 3$ and we consider $w = 1$. In all cases the cost of applying a meme is accounted as a fractional evaluation (i.e., as the fraction of the fitness function that needs being reevaluated as a result of a genotypic change) and added to the total number of evaluations. A run is terminated upon reaching 50,000 evaluations, and 20 runs are performed for each combination of problem, number of islands, ω_S and ω_R .

3.2 Experimental Results

First of all, full numerical results are provided in Table 1. As expected, the quality of results does globally improve when the number of islands is increased (note that all problems considered are maximization problems, and hence higher values are better). This is a well-known consequence of the use of decentralized evolutionary algorithms which naturally manifests itself in the multimemetic context as well. The main focus of this analysis is not how better results can get by increasing the number of islands though (an admittedly interesting issue that can be tackled in subsequent research), but the relative effect that design decisions regarding migrant selection and replacement have on the performance of the algorithm. To this end, we have conducted a systematic statistical analysis to ascertain the relative impact that each migration policy exerts on the iMMA.

We firstly consider results of all migrant selection strategies for either $\omega_R = \text{replace-worst}$ or $\omega_R = \text{replace-random}$. We perform a rank-based comparison by computing the relative ordering of each ω_S operator for a given problem and number of islands n_i : the selection strategy with the best mean is given rank 1 and the worst one is given rank 6 (recall there are six ω_S operators). In case of ties, the mean rank of the tied positions is awarded. Fig. 2 shows the distribution of ranks for each ω_S operator. Notice that these ranks are mostly consistent for both migrant replacement strategies. The fact that **random-immigrants** ranks consistently in the last positions is compatible with the fact that the iMMA is actually benefiting from the information exchange among islands beyond pure random diversity (hence the better results for a increasing number of islands – see Table 1). Also, **best** ranks in a poor position in both cases. This is often attributed to the premature convergence induced by this more intensive strategy. Indeed, this effect is illustrated in Fig. 1 (left). Note in any case that higher global diversity per se does not equate to better performance: MMAs also require that memes sustainedly support the search process and strategies such as **random** and **diverse-meme** are better at this, see Fig. 1 (right). In fact, another interesting observation is that migrant selection strategies based on memetic diversity perform better than their genetic counterparts. This indicates that injecting new diverse memetic material can have a larger influence in the behavior of the algorithm than just new genetic material, in line with the active role that the former actually has on the search process itself. The **random** strategy provides

Table 1. Results (20 runs) of the different iMMAs on TRAP, HIFF, HXOR and SAT, using the **replace-worst** strategy (upper half) and the **replace-random** strategy (lower half). The median (\tilde{x}), mean (\bar{x}) and standard error of the mean ($\sigma_{\bar{x}}$) are shown.

	TRAP		HIFF		HXOR		SAT	
	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$	\tilde{x}	$\bar{x} \pm \sigma_{\bar{x}}$
replace-worst								
sMMA	$n_i = 1$	31.4 30.0 \pm 0.5	408.0	427.6 \pm 13.9	360.0	360.2 \pm 4.4	547.0	546.6 \pm 0.4
	$n_i = 2$	30.6 30.3 \pm 0.3	456.0	471.1 \pm 17.2	380.0	382.4 \pm 6.2	547.0	547.3 \pm 0.3
random	$n_i = 4$	31.2 30.6 \pm 0.4	520.0	509.2 \pm 15.7	408.0	416.0 \pm 10.5	548.0	547.5 \pm 0.4
	$n_i = 8$	31.6 30.7 \pm 0.4	576.0	564.0 \pm 8.3	412.0	415.1 \pm 7.3	547.0	547.5 \pm 0.3
best	$n_i = 2$	29.6 30.0 \pm 0.3	436.0	470.0 \pm 20.4	372.0	379.4 \pm 7.6	547.0	547.1 \pm 0.4
	$n_i = 4$	30.8 30.3 \pm 0.4	456.0	469.4 \pm 17.2	380.0	385.8 \pm 5.4	546.0	546.5 \pm 0.4
	$n_i = 8$	31.4 30.3 \pm 0.4	576.0	519.6 \pm 14.5	374.0	378.2 \pm 6.9	547.0	546.9 \pm 0.4
diverse-gene	$n_i = 2$	29.6 29.6 \pm 0.4	456.0	473.8 \pm 16.5	384.0	384.0 \pm 4.7	548.0	547.4 \pm 0.3
	$n_i = 4$	30.2 30.0 \pm 0.4	528.0	499.7 \pm 18.3	395.0	404.6 \pm 10.7	547.0	546.9 \pm 0.4
	$n_i = 8$	30.4 30.1 \pm 0.4	576.0	543.6 \pm 11.8	394.0	404.8 \pm 8.5	547.0	547.3 \pm 0.4
diverse-meme	$n_i = 2$	30.6 30.1 \pm 0.4	456.0	475.8 \pm 16.0	380.0	381.4 \pm 4.4	548.0	547.4 \pm 0.4
	$n_i = 4$	31.0 30.5 \pm 0.4	472.0	501.2 \pm 16.1	404.0	411.2 \pm 11.4	548.0	547.5 \pm 0.3
	$n_i = 8$	31.2 30.7 \pm 0.3	576.0	553.2 \pm 10.5	402.0	418.2 \pm 11.9	548.0	547.5 \pm 0.3
random-immigrants	$n_i = 2$	29.4 29.4 \pm 0.5	436.0	451.0 \pm 15.7	352.0	356.4 \pm 5.0	547.0	547.5 \pm 0.3
	$n_i = 4$	28.6 28.7 \pm 0.5	454.0	453.2 \pm 16.0	348.0	351.6 \pm 3.6	547.0	546.9 \pm 0.3
	$n_i = 8$	30.4 29.3 \pm 0.5	454.0	471.1 \pm 14.8	331.0	336.9 \pm 4.3	547.0	546.8 \pm 0.3
probabilistic	$n_i = 2$	31.4 30.5 \pm 0.4	456.0	493.6 \pm 17.6	374.0	386.2 \pm 7.2	547.0	547.0 \pm 0.3
	$n_i = 4$	32.0 30.8 \pm 0.4	464.0	500.0 \pm 13.6	394.0	387.6 \pm 5.0	548.0	547.4 \pm 0.3
	$n_i = 8$	32.0 30.4 \pm 0.5	576.0	551.6 \pm 11.3	390.0	390.2 \pm 3.9	547.0	547.4 \pm 0.3
replace-random								
sMMA	$n_i = 1$	31.4 30.0 \pm 0.5	408.0	427.6 \pm 13.9	360.0	360.2 \pm 4.4	547.0	546.6 \pm 0.4
	$n_i = 2$	30.4 30.1 \pm 0.3	456.0	480.0 \pm 18.9	380.0	390.4 \pm 7.5	547.0	547.0 \pm 0.4
random	$n_i = 4$	30.8 30.2 \pm 0.4	520.0	506.8 \pm 16.5	412.0	426.6 \pm 10.0	547.0	547.2 \pm 0.3
	$n_i = 8$	31.4 30.7 \pm 0.3	576.0	543.2 \pm 11.6	408.0	414.9 \pm 6.4	548.0	547.5 \pm 0.3
best	$n_i = 2$	29.6 30.0 \pm 0.4	440.0	457.2 \pm 19.3	378.0	379.2 \pm 5.8	547.0	547.1 \pm 0.3
	$n_i = 4$	31.0 30.3 \pm 0.4	468.0	482.8 \pm 20.5	378.0	390.6 \pm 7.8	547.5	547.0 \pm 0.5
	$n_i = 8$	31.4 30.2 \pm 0.5	576.0	518.8 \pm 14.8	384.5	384.6 \pm 6.6	547.0	547.1 \pm 0.3
diverse-gene	$n_i = 2$	30.6 30.2 \pm 0.4	456.0	471.0 \pm 17.4	384.0	385.7 \pm 8.1	547.0	546.7 \pm 0.3
	$n_i = 4$	30.0 29.7 \pm 0.4	520.0	503.8 \pm 17.1	385.0	391.9 \pm 7.7	547.0	546.8 \pm 0.4
	$n_i = 8$	31.6 30.4 \pm 0.4	576.0	546.8 \pm 11.6	382.0	388.5 \pm 5.7	547.0	546.6 \pm 0.3
diverse-meme	$n_i = 2$	31.0 30.2 \pm 0.4	464.0	490.6 \pm 18.6	388.0	386.0 \pm 6.6	547.0	547.2 \pm 0.3
	$n_i = 4$	30.6 30.5 \pm 0.3	472.0	503.2 \pm 14.1	396.0	405.1 \pm 5.9	548.0	547.6 \pm 0.3
	$n_i = 8$	32.0 31.1 \pm 0.3	576.0	535.0 \pm 12.8	408.0	411.9 \pm 9.4	547.0	547.3 \pm 0.3
random-immigrants	$n_i = 2$	30.6 29.8 \pm 0.5	432.0	442.4 \pm 15.1	352.0	355.6 \pm 3.9	546.5	546.9 \pm 0.4
	$n_i = 4$	28.4 28.8 \pm 0.5	456.0	446.3 \pm 19.6	352.0	353.9 \pm 2.9	548.0	547.3 \pm 0.4
	$n_i = 8$	28.6 29.1 \pm 0.5	456.0	475.8 \pm 16.4	338.0	337.8 \pm 4.1	547.0	547.0 \pm 0.4
probabilistic	$n_i = 2$	30.9 30.5 \pm 0.4	464.0	482.4 \pm 16.8	370.0	372.4 \pm 5.1	547.0	546.7 \pm 0.4
	$n_i = 4$	31.6 30.9 \pm 0.3	576.0	518.6 \pm 15.5	384.0	384.6 \pm 4.9	547.0	547.3 \pm 0.3
	$n_i = 8$	31.0 30.3 \pm 0.5	576.0	548.4 \pm 13.1	394.0	400.9 \pm 5.8	547.0	547.3 \pm 0.3

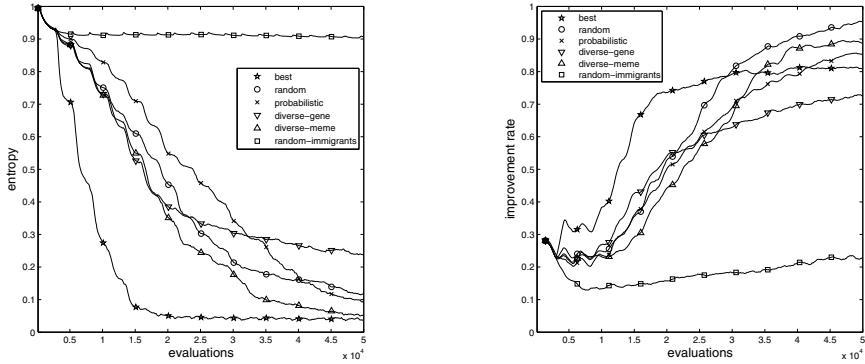


Fig. 1. (Left) Global population entropy (Right) Meme success rate (percentage of meme applications that result in an improvement). In both cases the data corresponds to HIFF, $n_i = 8$, $\omega_R = \text{replace-worst}$.

Table 2. Statistical tests for the migration selection strategies ($\alpha = 0.05$)

	Critical value	replace-worst	replace-random
Friedman	11.070498	35.416667	28.238095
Quade	2.382823	14.811603	9.189948

high performance as well, which could be attributed to its constituting a good tradeoff between genetic and memetic diversity. Along this line, note that this selection strategy performs slightly better in a relative sense when used in conjunction with **replace-worst** due to its more exploratory nature being compensated by the more intensive character of the latter replacement strategy. Note finally how the probabilistic generation of migrants sits comfortably in the third position in either case, not far from **random** selection. Obviously, a simple univariate model cannot adequately grasp the interdependencies among variables and hence this strategy behaves as a more exploratory variation of the **random** strategy.

To determine the extent to which rank differences are significant we use two well-known non-parametric statistical tests, namely Friedman and Quade tests. The results, at the standard level of $\alpha = 0.05$, are shown in Table 2. The statistic values obtained are clearly higher than the critical ones, thus indicating significant differences in their ranks. Hence, we have performed a post-hoc analysis using Holm test to determine whether the differences are significant with respect to a control strategy (in this case the strategy which provided the best average rank as shown in Fig. 2). Table 3 shows the results of this test. Notice that the test is passed in either case for **random-immigrants**, **best** and **diverse-gene**, hence indicating the control algorithm is significantly better than these. No statistical differences can be shown between **diverse-meme**, **random** and **probabilistic**.

If an analysis is conducted along the replacement dimension, i.e., by keeping fixed the selection strategy and comparing both replacement strategies, we can

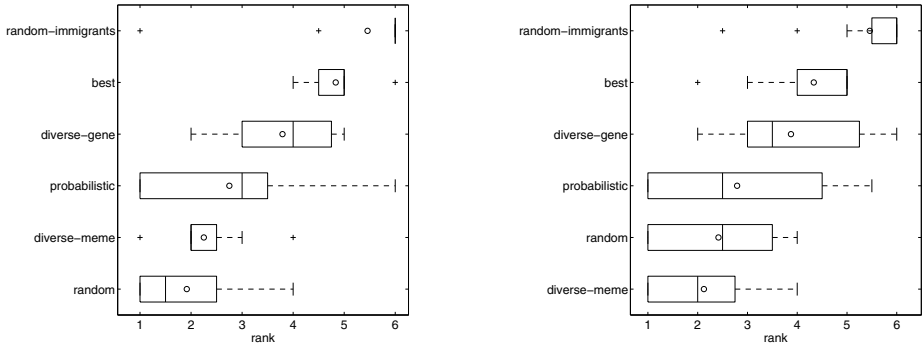


Fig. 2. Rank distribution of migration selection strategies. Each box comprises from the first to the third quartile of the distribution, the median (2nd quartile) is marked with a vertical line, the mean with a circle, whiskers span 1.5 times the inter-quartile range, and outliers are indicated with a plus sign. (Left) Results for $\omega_R = \text{replace-worst}$ (Right) Results for $\omega_R = \text{replace-random}$.

Table 3. Results of Holm test. (Top) $\omega_R = \text{replace-worst}$ using $\omega_S = \text{random}$ as control strategy (Bottom) $\omega_R = \text{replace-random}$ using $\omega_S = \text{diverse-meme}$ as control strategy.

	i	strategy	z -statistic	p -value	α/i
replace-worst	1	diverse-meme	0.436436	0.331260	0.050000
	2	probabilistic	1.091089	0.137617	0.025000
	3	diverse-gene	2.454951	0.007045	0.016667
	4	best	3.818813	0.000067	0.012500
	5	random-immigrants	4.637130	0.000002	0.010000
replace-random	1	random	0.381881	0.351275	0.050000
	2	probabilistic	0.872872	0.191367	0.025000
	3	diverse-gene	2.291288	0.010973	0.016667
	4	best	2.891387	0.001918	0.012500
	5	random-immigrants	4.364358	0.000006	0.010000

observe that `replace-worst` performs slightly better than `replace-random` but the difference does not reach significance at 0.05 level in any case, using a Wilcoxon ranksum test to perform head-to-head comparisons between both replacement strategies in each (problem, ω_S , n_i) combination. If we analyze specific (ω_S, ω_R) pairs, we find that there are statistically significant differences in the rank distribution of the 12 combinations (using Friedman and Quade test: values of 71.317308 and 11.251898 are respectively obtained, much larger than the critical values 19.675138 and 1.868615). Holm test is subsequently performed as shown in Table 4. Consistently with the previous results, the test is passed for all pairs involving `random-immigrants`, `best` and `diverse-gene` using `random+replace-worst` as control algorithm. No statistical differences can be shown between pairs involving `diverse-meme`, `random` and `probabilistic`.

Table 4. Results of Holm Test for all combinations of selection/replacement strategies, using random+replace-worst as control strategy

i	strategy	z -statistic	p -value	α/i
1	diverse-meme+replace-worst	0.651059	0.257504	0.050000
2	diverse-meme+replace-random	0.735980	0.230871	0.025000
3	probabilistic+replace-worst	1.075663	0.141039	0.016667
4	random+replace-random	1.103970	0.134803	0.012500
5	probabilistic+replace-random	1.755029	0.039627	0.010000
6	diverse-gene+replace-worst	2.745772	0.003018	0.008333
7	diverse-gene+replace-random	3.085455	0.001016	0.007143
8	best+replace-random	3.623287	0.000145	0.006250
9	best+replace-worst	4.132811	0.000018	0.005556
10	random-immigrants+replace-worst	5.123554	0.000000	0.005000
11	random-immigrants+replace-random	5.180167	0.000000	0.004545

4 Conclusions

The choice of migrant selection and migrant replacement operators is acknowledged as having a crucial impact on the performance of island-based GAs. In this work we have conducted an analysis of the influence of these two operators in the context of MMAs, in which individuals are not just points in the search space but also carry information on how to perform the search. Besides confirming some results which had been previously reported in the context of GAs (such as, e.g., the fact a migrating the best individual leads to a quick degradation of diversity and diminished performance), we have found that the replacement strategy (at least in the two incarnations considered) has less impact than the selection strategy, and that a selection strategy purely aimed at maintaining genotypic diversity does not compare favorably to other strategies based on memetic diversity (although the former still provides better results than a single-island panmictic model or a strategy based on random immigrants). The latter performs statistically similar to two other strategies aimed at randomly sampling the emitting population. It is specifically interesting to note that a probabilistic modeling of the population (even an arguably simple one such as the univariate model considered here) is still competitive with other migration operators. This suggests a potential line of future developments focusing on more complex probabilistic models capturing bivariate or multivariate dependencies [8]. Confirming these findings on problem instances of higher dimensionality and on other self-adaptive memetic models [15] are other interesting lines of future work.

Acknowledgements. This work is supported by MICINN project ANYSELF (TIN2011-28627-C04-01), by Junta de Andalucía project DNEMESIS (P10-TIC-6083) and by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

References

1. Alba, E., Troya, J.M.: Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. *Appl. Intell.* 12(3), 163–181 (2000)
2. Araujo, L., Merelo Guervós, J.J.: Multikulti algorithm: Using genotypic differences in adaptive distributed evolutionary algorithm migration policies. In: *IEEE Congress on Evolutionary Computation*, pp. 2858–2865. IEEE, Trondheim (2009)
3. Cantú-Paz, E.: Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics* 7(4), 311–334 (2001)
4. Cheng, J., Zhang, G., Neri, F.: Enhancing distributed differential evolution with multicultural migration for global numerical optimization. *Information Sciences* 247, 72–93 (2013)
5. De Falco, I., Della Cioppa, A., Maisto, D., Scafuri, U., Tarantino, E.: Biological invasion-inspired migration in distributed evolutionary algorithms. *Information Sciences* 207, 50–65 (2012)
6. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) *Second Workshop on Foundations of Genetic Algorithms*, pp. 93–108. Morgan Kaufmann, Vail (1993)
7. Grefenstette, J.: Genetic algorithms for changing environments. In: Männer, R., Manderick, B. (eds.) *Parallel Problem Solving from Nature II*, pp. 137–144. Elsevier, Brussels (1992)
8. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation* 1(3), 111–128 (2011)
9. Krasnogor, N., Blackburne, B.P., Burke, E.K., Hirst, J.D.: Multimeme algorithms for protein structure prediction. 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. 769–778. Springer, Heidelberg (2002)
10. Moscato, P.: Memetic algorithms: A short introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) *New Ideas in Optimization*. McGraw-Hill's Advanced Topics In Computer Science Series, pp. 219–234. McGraw-Hill, London (1999)
11. Neri, F., Cotta, C., Moscato, P.: *Handbook of Memetic Algorithms*. SCI, vol. 379. Springer, Heidelberg (2012)
12. Nogueras, R., Cotta, C.: Analyzing meme propagation in multimemetic algorithms: Initial investigations. In: *2013 Federated Conference on Computer Science and Information Systems*, pp. 1013–1019. IEEE Press, Cracow (2013)
13. Ong, Y.S., Lim, M.H., Chen, X.: Memetic computation—past, present and future. *IEEE Computational Intelligence Magazine* 5(2), 24–31 (2010)
14. Skolicki, K., Jong, K.D.: The influence of migration sizes and intervals on island models. In: *Genetic and Evolutionary Computation Conference 2005*, pp. 1295–1302. ACM, New York (2005)
15. Smith, J.: Self-adaptative and coevolving memetic algorithms. In: Neri, F., Cotta, C., Moscato, P. (eds.) *Handbook of Memetic Algorithms*. SCI, vol. 379, pp. 167–188. Springer, Heidelberg (2012)
16. Watson, R.A., Pollack, J.B.: Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In: *1999 IEEE Congress on Evolutionary Computation*, pp. 292–297. IEEE Press, Washington D.C (1999)
17. Weber, M., Neri, F., Tirronen, V.: A study on scale factor in distributed differential evolution. *Information Sciences* 181(12), 2488–2511 (2011)

Tuning Evolutionary Multiobjective Optimization for Closed-Loop Estimation of Chromatographic Operating Conditions

Richard Allmendinger, Spyridon Gerontas,
Nigel J. Titchener-Hooker, and Suzanne S. Farid

Department of Biochemical Engineering, University College London,
Torrington Place, London WC1E 7JE, UK
{r.allmendinger,s.gerontas,nigelth,s.farid}@ucl.ac.uk

Abstract. Purification is an essential step in the production of biopharmaceuticals. Resources are usually limited during development to make a full assessment of operating conditions for a given purification process commonly consisting of two or more chromatographic steps. This study proposes the optimization of all operating conditions simultaneously using an evolutionary multiobjective optimization algorithm (EMOA). After formulating the closed-loop optimization problem, which is subject to constraints and resourcing issues, four state-of-the-art EMOAs — NSGAI, MOEA/D, SMS-EMOA, and ParEGO — were tuned and evaluated on test problems created from real-world data available in the literature. The simulation results revealed that the performance of an EMOA depends on the setting of the population size, and constraint and resourcing issue-handling strategies adopted. Tuning these algorithm parameters revealed that the EMOAs, in particular SMS-EMOA and ParEGO, are able to discover reliably within 100 evaluations operating conditions that lead to high levels of yield and product purity.

1 Introduction

Manufacturing costs of therapeutic proteins are driven by costs associated with the purification of a protein of interest from impurities, such as host cell proteins and DNA, arising during the fermentation and harvest process, and by the need to achieve strictly controlled levels of key impurities. Chromatography is a commonly-used technique for purifying proteins and has been identified as a key cost driver [1]. The overall goal of this study is to optimize the operating conditions of a chromatography platform so as to improve multiple criteria, such as recovery yield and final product purity, contributing to a reduction in manufacturing costs.

Approaches for optimizing a chromatographic process can be classified broadly into two classes: *experimentally-validated simulation approaches* or *direct experimental approaches* [2]. The former includes approaches that describe a chromatographic process using a (predictive) linear or non-linear model based on mass transfer and thermodynamics [3]. Although simulation-based, this approach relies on physical experiments being performed to calibrate and validate the model. Various optimization methods have been used to estimate the parameters of a chromatographic model (see e.g. [3,4]).

If the process modeled is not well-understood or cost prohibitive to define in terms of a simulation model, then a *direct experimental optimization approach* can be adopted. Such approaches optimize a chromatographic process by performing physical experiments guided, for example, by *design of experiments* (DoE) in combination with a response surface analysis [5,6] or an evolutionary algorithm (EA) [2,7]. An experimental optimization approach might incur high experimental costs, whilst a simulation-based approach relies heavily on the computational resources available.

Recently, multiobjective methods have found application in chromatography processes optimization. For example, in [8] an evolutionary multiobjective optimization algorithm (EMOA) was used within a simulation-based approach to optimize purity, productivity, and/or yield of a single chromatography step. EMOAs have found application in various experimental optimization problems [9], and are easily adaptable to problems featuring constrained, non-linear, non-convex, noisy, dynamically changing, and/or multiple objective functions. EMOAs have also been extended to cope with resourcing issues in experimental optimization leading e.g. to delayed/missing objective values [10] and temporary non-availability of certain solutions for evaluation [9]. The issue around missing objective values can also be encountered in chromatography process optimization and is investigated in more detail in this study.

Although a chromatographic purification process consists of multiple steps, the work cited above focuses on the optimization of a single step only. The goal of this study is to optimize multiple chromatography steps simultaneously so as to maximize recovery yield and final product purity (or equivalently minimize impurities). Optimizing multiple steps means that ideally interactions, technical limitations and/or resourcing issues between steps can be accounted for in the optimization. The lack of models capable of capturing interactions and constraints between multiple chromatography steps accurately, means however that a direct experimental approach needs to be adopted. To realize this experimental optimization platform, a sophisticated and precise laboratory setup is required as well as an optimization method capable of dealing efficiently with the enlarged search space and additional constraints (arising due to the presence of multiple chromatography steps). This study focuses on the design of an effective optimizer to guide the selection of conditions for physical experiments.

2 Problem Definition

This section describes the problem formulation for the multiobjective optimization of chromatographic operating conditions (MOCOC) subject to resourcing issues. The experimental platform adopted for the optimization of operating conditions across multiple chromatography steps is visualized in Figure 1, and can be formulated mathematically as follows:

$$\begin{aligned} &\text{maximize } \mathbf{f}(\mathbf{x}, \sigma) = (f_1(\mathbf{x}, \sigma), \dots, f_m(\mathbf{x}, \sigma)) \\ &\text{subject to } \mathbf{x} \in X, \end{aligned}$$

where $\mathbf{x} = (x_1, \dots, x_l)$ is a *solution vector* (here a set of operating conditions), and X a *feasible search space* (here the set of all possible operating conditions). The *objective*

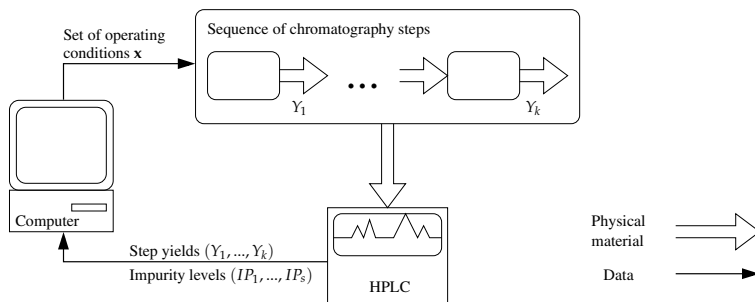


Fig. 1. Schematic of a typical experimental setup for the closed-loop optimization of chromatographic operating conditions. Following the set up of the operating conditions, defined by \mathbf{x} , the sample is passed through a sequence of chromatography steps $i = 1, \dots, k$. An HPLC device is used to obtain the step yields Y_i and the final levels of individual impurities $IP_j, j = 1, \dots, s$. Based on this quality measure, an optimizer running on the computer then selects the next set of operating conditions for testing.

vector function f is a *black box* and represents a time-consuming and costly physical experiment on \mathbf{x} , which is characterized by $m > 0$ noisy measurements f_1, \dots, f_m . The functions f_i are known as *objectives* and are typically in conflict. The vector σ represents environmental factors that cannot be controlled, e.g. imprecision in the experimental equipment. In the following, these problem features are described in more detail.

Decision variables x_1, \dots, x_l : A solution vector \mathbf{x} represents a set of relevant operating conditions, such as pH and salt concentration, for a set of chromatography steps $i = 1, \dots, k$. Figure 2 shows the solution encoding used in this work: each step i is associated with a pre-defined resin i and a variable number of operating conditions $c_{i,j}, 1 \leq j \leq d_i$, resulting in $l = \sum_{i=1}^k d_i$ decision variables in total. Typically, the values $c_{i,j}$ are represented by discretized real values.

Objective functions f_1, \dots, f_m : Two ($m = 2$) commonly-used metrics were considered in order to characterize the quality of a chromatographic process: the overall recovery yield Y and the final product impurities $\sum IP_j$:

$$\begin{aligned} \text{maximize } f_1 &= Y = Y_1 \times \dots \times Y_k \\ \text{minimize } f_2 &= \sum IP_j = \sum_{j=1}^s IP_j, \end{aligned}$$

where Y_i is the yield of chromatography step i , and $IP_j, j = 1, \dots, s$, the levels of different impurity types, such as host cell proteins and DNA; note, the objective of minimizing product impurities is equivalent to maximizing purity and used here due to the structure of the test problems considered (see Section 3.1). Both objectives, yield and (im)purity, are obtained by analyzing the sample using an HPLC (high-performance liquid chromatography) device. Measuring the yield and levels of different impurities takes around 4 min and 30 min per sample, respectively, whilst the robot takes around 1

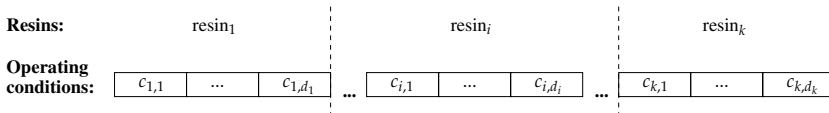


Fig. 2. Representation of a solution \mathbf{x} for a process with $i = 1, \dots, k$ chromatography steps. Each step i is linked to a fixed resin i and a set of operating conditions $c_{i,j}, 1 \leq j \leq d_i$.

hour per chromatography step to prepare a sample (this robotic step can be parallelized to up to 8 samples).

Feasible Search Space X : In addition to standard constraints on the decision variable value ranges, the search space may be defined by *dependency constraints* between chromatography steps. For instance, setting the salt concentration of a step i dictates the lowest possible salt concentration of the successive step, or $c_{i,j} \leq c_{i+1,t}$, assuming indices j and t point to the salt concentration at chromatography step i and $i + 1$, respectively.

In addition to constraints defining the search space X , there may also be constraints on the objective values f_1, \dots, f_m . For example, for antibodies, regulatory requirements specify that the final product impurities needs to be $\sum IP_j < 5\%$. Moreover, in the presence of limited resources, there may be a threshold Y_{\min} on the minimum recovery yield. This limitation can be seen as a *resourcing issue* and shall prevent the waste of resources dedicated to the evaluation of inefficient purification processes. Mathematically, this resourcing issue can be expressed by a Boolean clause as follows

$$\text{if } Y_i \times \dots \times Y_p < Y_{\min} \text{ then terminate experiment and return } Y_1, \dots, Y_p \quad (1)$$

where $p \leq k$ denotes the chromatography step after which the cumulative yield is below the threshold Y_{\min} . That is, if the resourcing issue is ‘activated’, then the objectives Y and $\sum IP_j$ are missing. However, the measurements Y_1, \dots, Y_p are available and might be used to estimate Y and/or $\sum IP_j$.

Uncertainties: The decision variables x_1, \dots, x_l and the measurements f_1 and f_2 might be subject to some level of uncertainty (noise) given the experimental nature of the problem. This level is typically low if the experimental platform is set up accurately, and thus neglected here.

3 Experimental Setup

This section describes the case study, extensions augmented on the EMOAs for coping with the challenges of the MOCOC problem, and algorithm parameter settings as used in the subsequent experimental analysis.

3.1 Case Study

Ultimately, the goal is to tackle MOCOC problems with $k \approx 3$ chromatography steps and $l \approx 8$ operating conditions in total. The purification sequence considered in [6] falls into this problem domain and was used in this study as the “test problem” to tune and validate different state-of-the-art EMOAs (using computational experiments).

Table 1. MOCOC problem characteristics

Chromat. Step i	Operating condition	Value range	Step size δ
$i = 1$, affinity	pH _{wash}	[4.5; 5.5]	0.1
	pH _{elution}	[2.5; 3.5]	0.1
	NaCl _{wash}	[50; 500]	25
$i = 2$, cation exchange	pH _{load,2}	[4.5; 5.5]	0.1
	Grad. length Load	[10; 20] [45;55]	2 1
$i = 3$, anion exchange	pH _{load,3}	[7; 8]	0.1
	Load	[100; 200]	5

Table 2. EMOA default parameter settings

EMOA	Parameter	Setting
All	Max evaluations G	100
	Crossover probability p_c	0.6
	Per-variable mutation probability p_m	$1/l$
NSGAI	Population size n	10
MOEA/D	Population size n	20
	#Weight vectors T	20
SMS-EMOA	Population size μ	4
ParEGO	Initial population size n	50
	#Scalar vectors s	10

Table 1 lists the operating conditions to be optimized for each of the $k = 3$ steps (affinity, cation and anion exchange); the operating condition values were discretized as specified by the step size δ . The heatmap data published in [6] was used to construct two interpolated fitness landscapes for each of the $k = 3$ steps, one for the step yield Y_i and one for the step’s impurity level IP_i , using the Kriging approach.¹ The experimental study considered the problem with $k = 2$ steps (the first two steps) and $l = 6$ operating conditions, and the complete problem with all $k = 3$ steps and $l = 8$ operating conditions.

The independent optimization of each chromatography step has been studied before [6]. In our work the problem was extended by dependency constraints and resourcing issues (mimicking real limitations of the problems of interest): the dependency constraint was defined by $\text{pH}_{\text{wash}} \leq \text{pH}_{\text{load},2}$ (for the sake of this constraint, the value ranges of both variables were set identically), and the resourcing issue was represented by Equation (1).

3.2 Tuning Evolutionary Search for the MOCOC Problem

To run an EMOA on the MOCOC problem, strategies for coping with the constraints and resourcing issues need to be defined.

Handling Dependency Constraints: Four strategies — *random*, *copy*, *swap*, and *regenerate* — were investigated for coping with the dependency constraint defined above. Upon encountering an infeasible solution, the strategy *random* sets its $\text{pH}_{\text{load},2}$ value to a random value selected from the range $[\text{pH}_{\text{wash}}, 5.5]$, whilst the strategy *copy* sets $\text{pH}_{\text{load},2} = \text{pH}_{\text{wash}}$. The strategy, *swap*, swaps the values of $\text{pH}_{\text{load},2}$ and pH_{wash} , which results in a feasible solution due to the identical value range of the two variables. Finally, the strategy *regenerate* iteratively generates new solutions until it generates one that is feasible.

Handling Resourcing Issues: Three strategies — *strict penalizing*, *relaxed penalizing*, and *fitness-inheritance* — were investigated for coping with the resourcing issue defined in Equation (1). The aim of these strategies is to substitute missing objective

¹ A Kriging function, $\text{Krig}()$, was used from the *fields* package of the statistical software R.

values with some surrogate. The strategy, *strict penalizing*, sets the objectives of a solution violating Equation (1) to the worst possible values; i.e. $f_1 = 0 = 1$ (assuming a normalized objective space). The strategy, *relaxed penalizing*, uses the available yield measurements to set the objective values to $f_1 = Y_i \times \dots \times Y_p$ and $f_2 = 0$. Finally, the strategy, *fitness-inheritance*, selects for a solution with missing objectives a solution from the set of all solutions evaluated so far that is both closest to it in the decision space (in terms of normalized Euclidean distance) and has no missing objectives, and then simply copies the solution's values of f_2 and Y_{p+1}, \dots, Y_k to allow the computation of f_1 .

3.3 Algorithm Parameter Settings

Four state-of-the-art EMOAs were considered in the experimental analysis: NSGAI [11], MOEA/D [12], SMS-EMOA [13], and ParEGO [14]. All EMOAs avoided the evaluation of duplicate solutions (solutions were regenerated until a unique one is created), each used a latin hypercube initialization procedure, uniform crossover, binary tournament selection (with replacement), and a mutation operator that selects a value at random from the feasible variable value range (see Table 1). Both NSGAI and MOEA/D employ a generational reproduction scheme (using a fixed population size of μ), whilst SMS-EMOA uses a steady-state scheme, and ParEGO considers all solutions evaluated to create a single solution.

The aim of the experimental study was to understand how the performance of the EMOAs is affected by different algorithm parameter settings, and constraint and resourcing issue-handling strategies. The default settings of the EMOAs are given in Table 2. The total number of evaluations $G = 100$ represents the estimated budget available for the problems of interest. Results shown are the average across 30 independent runs. Hypervolume and attainment surface results were obtained by considering all solutions found during a run that had no missing objective values. For the hypervolume calculation, the objective values were normalized to lie in the range $[0,1]$, and the reference point was set to a value of 2 for all objectives.

4 Experimental Analysis

The first two sets of experiments investigate the performance of the constraint-handling strategies and sensitivity of algorithm parameter settings in the absence of the resourcing issue, which is the focus of the last set of experiments.

Investigation of Constraint-Handling Strategies: Figure 3 shows the performance of the different constraint-handling strategies when augmented on NSGAI. From Figure 3(a) it is apparent there is a trade-off between the population size n and the number of generations G/n available for optimization: the performance increases until a population of $n \approx 10$ after which any further increases in n lead to a performance reduction (due to the smaller number of generations available). The *random* strategy performs most robustly, followed closely by the *copy* and *regenerate* strategy. The *swap* strategy performs significantly worse than the other strategies, in particular around the sweet spot of $n \approx 10$, as it deteriorates the original solution's string most. The attainment

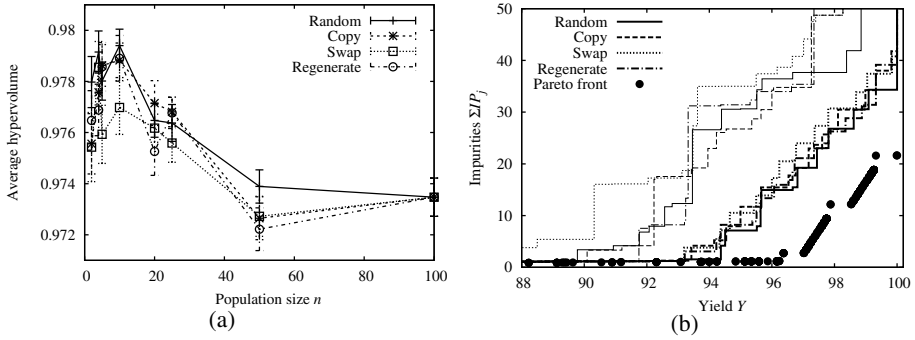


Fig. 3. a) Average hypervolume and its standard error as a function of the population size n , and b) worst (thin lines) and median (bold lines) attainment surfaces for $n = 10$ obtained by different constraint-handling strategies augmented on NSGAI for a MOCOC problem with $k = 2$ steps and $l = 6$ variables. For every setting marked by a point in a), a Kruskal-Wallis test (significance level of 5%) has been carried out. *Random*, *copy*, and *regenerate* perform best for $n = 10$. There is no clear winner for the other settings.

surface plot, Figure 3(b), confirms this ranking as well as the significant gap of the *random* strategy to the estimated true Pareto front (which has been obtained by taking the non-dominated front discovered across multiple and long runs of NSGAI). The performance patterns were similar for both test problems, i.e. $k = 2$ and 3 steps, and the other EMOAs. For ParEGO, the choice of the constraint-handling strategy is not as crucial as the search for a new solution is performed over an interpolated landscape (instead of the actual search space), which is cheap to evaluate.

Investigation of Crucial Algorithm Parameter Settings: Figure 4 analyses the sensitivity in performance of the different EMOAs as a function of algorithm parameter settings, in particular the population size n . From Figure 4(a), it is obvious that the performance of all EMOAs improves until a certain n is reached, and then degrades for further increases in n . SMS-EMOA is able to achieve the highest average hypervolume, when used in combination with small population size of $n \approx 4$. ParEGO performed slightly worse than SMS-EMOA in terms of the average hypervolume but is more robust to variations in n . Note, for $n = 100$, no optimization was performed as all EMOAs sample the search space using a latin hypercube design, which can be seen as the default performance obtained with a DoE approach. As can be seen from Figure 4(a), this performance is clearly beaten by the EMOAs for most settings of n . The performance ranking of EMOAs with respect to worst and median attainment surfaces obtained, which are shown in Figure 4(b), was in alignment with the hypervolume results. It is also apparent from the plot that SMS-EMOA and ParEGO were able to get significantly closer to the Pareto front than NSGAI.

Investigation of Resourcing Issue-Handling Strategies: Finally, Figure 5 analyzes the performance of the resourcing issue-handling strategies when augmented on SMS-EMOA. In Figures 5(a) and 5(b), the resourcing issue was present from the very beginning of the optimization, whilst, in Figures 5(c) and 5(d), it was ignored and the evaluation completed for the first 10 solutions with a cumulative yield below Y_{\min} .

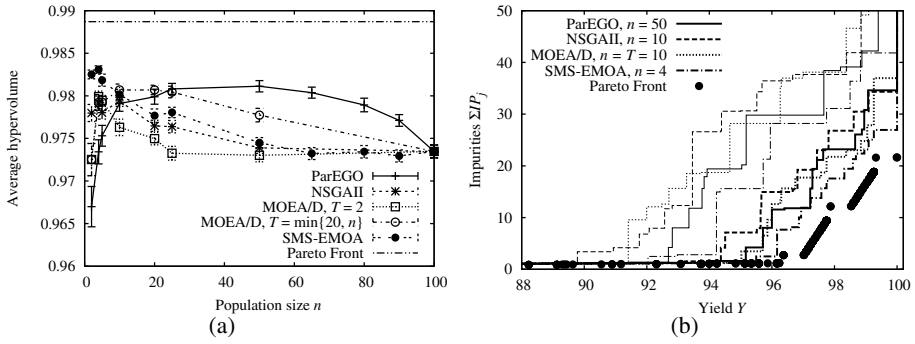


Fig. 4. a) Average hypervolume and its standard error as a function of the population size n , and b) worst (thin lines) and median (bold lines) attainment surfaces obtained for optimal settings of n by several EMOAs for a MOCOC problem with $k = 2$ steps and $l = 6$ variables. For every setting marked by a point in a), a Kruskal-Wallis test (significance level of 5%) has been carried out. SMS-EMOA performs best for $n < 10$, and ParEGO for $25 < n < 100$. There is no clear winner for the other settings.

From Figures 5(a) and 5(c) it can be observed that the presence of the resourcing issue has a significant negative impact on performance for $Y_{\min} > 90\%$. Relaxing the resourcing issue reduces the impact on performance but it is an expensive approach. Preventing the optimizer from entering certain regions of the objective space introduces a search bias towards other parts of the Pareto front, as evident from the attainment surfaces shown in Figures 5(b) and 5(d), especially in Figure 5(b), for $Y_{\min} = 94\%$. Comparing the different resource issue-handling strategies, it is apparent that a penalizing strategy performs best in Figures 5(a) and 5(b). A fitness-inheritance strategy performs better in the relaxed scenario (Figures 5(c) and 5(d)) because, once the resourcing issue is switched on, it allows the optimizer to enter more quickly a feasible region in the objective space than the penalizing strategies (as indicated by the number of experiments below Y_{\min}). Note, although relaxing the resourcing issue leads to more distributed attainment surfaces (see Figure 5(d)), the surfaces are further away from the Pareto front than in the unrelaxed case due to the lower level of exploitation. The other EMOAs are affected in a similar way by the resourcing issue.

5 Conclusion and Future Work

This paper has considered a real-world problem concerned with the optimization of operating conditions for chromatographic purification processes so as to maximize recovery yield and product purity. The problem has been formulated as a multiobjective closed-loop optimization problem subject to dependency constraints, resourcing issues, uncertainties, and a limited number of evaluations. Several strategies were proposed for dealing with the constraints and resourcing issues, and subsequently augmented and validated on four state-of-the-art EMOAs — ParEGO, NSGAI, MOEA/D, and SMS-EMOA — for two test problems created from published real-world data. The experimental study revealed that EMOAs can achieve a better performance within 100

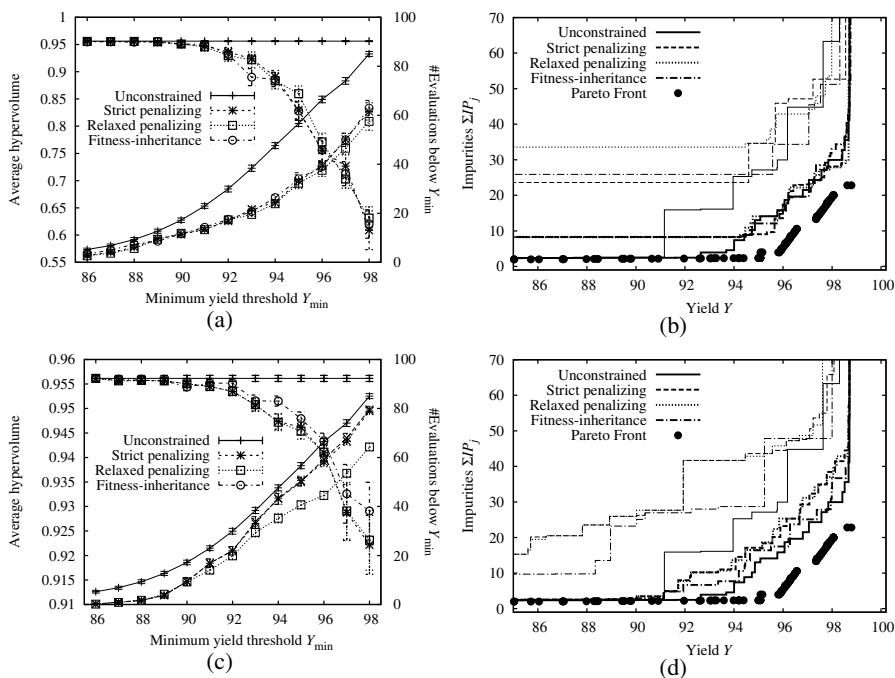


Fig. 5. a) and c) Average hypervolume, its standard error, and #evaluations below Y_{\min} as a function of the threshold Y_{\min} , and b) and d) worst (thin lines) and median (bold lines) attainment surfaces obtained for $Y_{\min} = 94\%$ by SMS-EMOA for a MOCOC problem with $k = 3$ steps and $l = 8$ variables. In a) and b), the resourcing issue was present throughout the search, whilst, in c) and d), it was ignored for the first 10 solutions with $f_1 < Y_{\min}$. For every setting marked by a point in a) and c), a Kruskal-Wallis test (significance level of 5%) has been carried out. Relaxed penalizing performs best in a) for $Y_{\min} = 95\%$, whilst fitness-inheritance performs best in c) for $Y_{\min} = 92\%$ and 94% . There is no clear winner for the other settings.

evaluations than a standard DoE approach, such as a latin hypercube design. The best performance was achieved by SMS-EMOA when used in combination with a small population of size $n \approx 4$ and a random sampling-based constraint-handling strategy. The performance of an EMOA depended on the resourcing issue-handling strategy: A penalizing strategy performed best if a resourcing issue is present throughout the search, whilst a fitness-inheritance approach performs better if the resourcing issue is relaxed. Future research will focus on applying SMS-EMOA to guide real physical chromatographic experiments.

References

1. Pollock, J., Bolton, G., Coffman, J., Ho, S.V., Bracewell, D.G., Farid, S.S.: Optimising the design and operation of semi-continuous affinity chromatography for clinical and commercial manufacture. *Journal of Chromatography A* 1284, 17–27 (2013)

2. Susanto, A., Treier, K., Knieps-Grünhagen, E., von Lieres, E., Hubbuch, J.: High throughput screening for the design and optimization of chromatographic processes: automated optimization of chromatographic phase systems. *Chemical Engineering & Technology* 32(1), 140–154 (2009)
3. Guiochon, G.: Preparative liquid chromatography. *Journal of Chromatography A* 965(1), 129–161 (2002)
4. Irizar Mesa, M., Llanes-Santiago, O., Herrera Fernández, F., Curbelo Rodríguez, C., Da Silva Neto, A.J., Câmara, L.D.T.: An approach to parameters estimation of a chromatography model using a clustering genetic algorithm based inverse model. *Soft Computing* 15(5), 963–973 (2011)
5. Ferreira, S.L.C., Bruns, R.E., Ferreira, H.S., Matos, G.D., David, J.M., Brandao, G.C., da Silva, E.G.P., Portugal, L.A., dos Reis, P.S., Souza, A.S., dos Santos, W.N.L.: Box-behnken design: An alternative for the optimization of analytical methods. *Analytica Chimica Acta* 597(2), 179–186 (2007)
6. GE Healthcare Life Sciences. A platform approach for the purification of antibody fragments (fabs). Application note 29-0320-66 AA (2012)
7. Treier, K., Berg, A., Diederich, P., Lang, K., Osberghaus, A., Dismer, F., Hubbuch, J.: Examination of a genetic algorithm for the application in high-throughput downstream process development. *Biotechnology Journal* 7, 1203–1215 (2012)
8. Nfor, B.K., Zuluaga, D.S., Verheijen, P.J.T., Verhaert, P.D.E.M., van der Wielen, L.A.M., Otens, M.: Model-based rational strategy for chromatographic resin selection. *Biotechnology Progress* 27(6), 1629–1643 (2001)
9. Allmendinger, R., Knowles, J.: On handling ephemeral resource constraints in evolutionary search. *Evolutionary Computation* 21(3), 497–531 (2013)
10. Allmendinger, R., Knowles, J.: ‘Hang On a Minute’: Investigations on the Effects of Delayed Objective Functions in Multiobjective Optimization. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 6–20. Springer, Heidelberg (2013)
11. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
12. Zhang, Q., Hui, L.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 11(6), 712–731 (2007)
13. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181(3), 1653–1669 (2007)
14. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* 10(1), 50–66 (2006)

A Geometrical Approach to the Incompatible Substructure Problem in Parallel Self-Assembly

Navneet Bhalla¹, Dhananjay Ipparthy², Eric Klemp³, and Marco Dorigo²

¹ Cornell University, Ithaca, New York, USA

navneet.bhalla@cornell.edu

² Université Libre de Bruxelles, Brussels, Belgium

{dhananjay.ipparthi,mdorigo}@ulb.ac.be

³ University of Paderborn, Paderborn, Germany

eric.klemp@uni-paderborn.de

Abstract. The inherent massive parallelism of self-assembly is one of its most appealing attributes for autonomous construction. One challenge in parallel self-assembly is to reduce the number of incompatible substructures that can occur in order to increase the yield in target structures. Early studies demonstrated how a simple approach to component design led components to self-assemble into incompatible substructures. Approaches have been proposed to reduce the number of incompatible substructures by increasing component complexity (e.g. using mechanical switches to determine substructure conformation). In this work, we show how a geometrical approach to self-assembling target structures from the inside-out eliminates incompatible substructures and increases yield. The advantages of this approach includes the simplicity of component design, and the incorporation of additional techniques to reduce component interaction errors. An experiment using millimeter-scale, 3D printed components is used to provide physical evidence to support our geometrical approach.

Keywords: Self-assembly, parallelism, yield, mesoscale, 3D printing.

1 Introduction

Self-assembly is prevalent throughout nature, and is the basis of construction for a myriad of complex biological structures [10]. Inspired by nature, self-assembly is viewed as an enabling technology for the creation of artificial systems [12]. Self-assembly is the autonomous organization of a set of components, in an environment, into structures without human intervention [16]. However, many aspects of self-assembly require further investigation in order to apply the advantages seen in nature to engineered systems, such as massive parallelism [9].

Parallelism in self-assembly can be exploited in two primary forms: (1) the parallel construction of a single target structure, versus (2) the parallel construction of multiple target structures. In both cases, emerging substructures may be incompatible due to binding mechanisms or geometry. In the latter case, a set

of components that can self-assemble into a single target structure may not be suitable for constructing multiple target structures due to newly introduced component interactions [2]. For example, an intuitive approach to designing a set of components that can self-assemble into a target hexagonal structure is to dissect the hexagon into fundamental, triangular components [7]. This simple approach is effective for self-assembling a single target structure in parallel. However, this leads to incompatible substructures (e.g. with two and five components) when constructing multiple target structures in parallel.

Furthermore, the latter case is connected to the yield problem, where the objective is to maximize the number of self-assembled target structures and minimize the amount of waste [12]. Reducing or eliminating the number of incompatible substructures that can occur during the self-assembly process is one approach to improving yield [7]. Here, we present a geometrical approach to the design of components that eliminates incompatible substructures from forming, by directing the self-assembly process to construct a target structure from the *inside-out*. The geometry of components in the context of parallel self-assembly has already been considered [11]. However, in this study the focus was not on the construction of target structures (with defined shape) and yield, but rather on the construction of arrays of components (without defined boundaries).

The following section presents the specific incompatible substructure problem of interest here, and methods for directing the self-assembly process to differentiate our geometrical approach. Next, an overview of our geometrical approach is provided, including the designs of components and their environment, in comparison to the simple approach. A description of an experiment follows, which uses millimeter-scale components fabricated by a 3D printer. Components are confined to the surface of a tray, and placed on an orbital shaker. Magnetism is used to attract and repel components. The results show a statistically significant difference between our geometrical approach and the simple approach, especially as the number of potential substructures grows. We discuss several areas of improvement to our approach as future work. We conclude by summarizing how this work provides physical evidence to support our geometrical approach to the incompatible substructure problem in parallel self-assembly.

2 Background

The seminal work by Hosokawa et al. [7] is used to introduce the incompatible substructure problem. Next, the proposed method for directing the self-assembly process by Hosokawa et al. to address this problem is described. Three additional methods to direct the process are presented to contrast our geometrical approach.

2.1 Incompatible Substructure Problem

An early work with the aim of analyzing the dynamics of self-assembling systems was presented by Hosokawa et al. [7]. The target structure for their study was a regular hexagon. The hexagon was divided into six equilateral triangles, serving

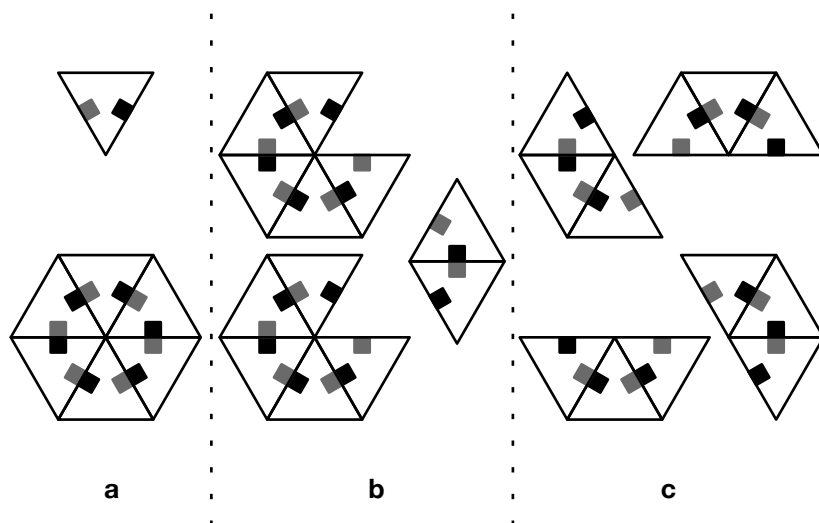


Fig. 1. A triangle component with magnetic north (black) and south (grey) and a target hexagon (a). An example of 12 components for 2 target structures in an intermediate state with incompatible substructures (b) versus compatible substructures (c).

as the base-shape for the centimeter-sized components in their system. Each mechanical component had two permanent magnets, with magnetic north and magnetic south facing outwards on two separate edges (Fig. 1).

With magnetism used to bind components, vibrational energy was used to mix the components. Components were confined to the surface of a rectangular box. The box was oriented vertically, and rotated perpendicularly to its major axis. Misaligned components, errors, could be corrected through component-environment interactions due to the vibrations and gravity.

Substructures consisted of one to five components. It is impossible to have incompatible substructures when constructing a single target structure, with the exact number of components. However, when constructing multiple target structures in parallel, incompatible substructures can occur (Fig. 1). Hosokawa et al. calculated the probability of the five types of substructures binding together [7]. Based on these probabilities, they were able to derive a master equation, conceptually similar to chemical kinetics. Their analytical method was used to both calculate yield and understand the dynamics of this system over time.

2.2 Directed Self-Assembly

To improve yield, Hosokawa et al. proposed an alternative set of components to the simple, homogenous set of components [7]. The alternative set exploited conformational switching, a mechanism that changes a feature (e.g. binding mechanism or geometry) of a component based on local component interaction [13].

The alternative set consisted of two types, *seed* and *variable*, components. One seed component is required for each target structure. Permanent magnets of opposite polarity were embedded within two edges of a seed component. However, variable components had two permanent magnets placed within their interior, which would only move to the edge when a magnet of opposite polarity was present in a neighboring component. The use of a seed and variable components created an *assembly sequence*, and eliminated the formation of incompatible substructures. One drawback was that the proposed seed components could self-assemble, resulting in errors (due to magnetic binding and the triangular shape of the components). Hosokawa et al. analytically showed how their conformational switching approach could improve yield over their simple approach [7]. To the best of our knowledge, this system has not been physically implemented.

Additional designs for conformational switching and alternative methods for directing the self-assembly process have been developed. Engineered proteins using ligand switches [4] and robotic modules using electro-mechanical switches [8] have been physically implemented. Synthesized DNA using a nucleic acid switch has been theorized [5]. DNA self-assembly using the abstract Tile Assembly Model, leverages seed tiles, environment temperature, and cooperative binding to direct crystallization [17]. Seed tiles with a larger number of binding sites have been physically demonstrated to improve the yield of algorithmic crystals [1]. Alternatively, the self-assembly process can be divided into time steps, and the set of components at different stages can also be used to direct the process by constructing target structures from the inside-out [2].

There are several drawbacks to these methods. For the staging method, it becomes an increasingly difficult task to prevent error interactions when increasing the number of components and interdependent stages. Permitting binding at one temperature and preventing binding at another is the biggest challenge in physically implementing systems based on the abstract Tile Assembly Model. Conformational switches are challenging to engineer, and as shown in the next section, unnecessary for self-assembling target structures with basic geometries.

3 Geometrical Approach

We present our geometrical approach to the incompatible substructure problem in parallel self-assembly. The core idea of our approach is that component geometry can direct the self-assembly process, as an alternative to conformational switching, multiple environmental conditions, or staging. A variation to the simple component set, by Hosokawa et al., is used to contrast a component set based on our geometrical approach. This change in geometry directs the target structure to form from the inside-out, instead of the less versatile assembly path generated by the conformational switching approach by Hosokawa et al.

Instead of a regular hexagon, a circle is used as the target structure. This change in target shape reduces the contact surface area between target structures, decreasing the potential for interaction errors and improving the mixing of target structures and substructures. It also allows for the target shape to remain

constant, while being able to vary the number of components with regularity. Scaling the number of divisions of the circle from three (minimum number for incompatible substructures to form) to six allows for deeper investigation.

Mesoscale (micrometer to millimeter) self-assembly offers flexibility in design and functionality that is unparalleled by molecular systems [15]. Millimeter-scale components, fabricated by a 3D printer, are used in this work. The target circle is 25 mm in diameter. Three types of components are used in this work, *sector* (first, homogenous set), and *disc* and *ray* (second, heterogeneous set).

Sector components are similar to the simple set used by Hosokoawa et al. (named after dividing a circle into equal subunits). Sector (S) components are denoted by C_x^S , where $x \in \{3, 4, 5, 6\}$. Instead of embedding two magnets of opposite polarity in the edges, one permanent magnet (polarity north) and a ferromagnetic micro-screw are used (Fig. 2). This reduces misalignment errors by up to 50%, in comparison to the simple component set by Hosokawa et al.

The design of disc and ray components are based on our geometrical approach (named after sunflowers; ray flowers around the middle disc flowers [14]). Disc (D) and ray (R) components are denoted by C_x^D and C_x^R , where $x \in \{3, 4, 5, 6\}$. A ferromagnetic micro-screw is used in each ray, and one permanent magnet (polarity north) is used in each ray location in a disc (Fig. 2). As well, *key-lock* shapes are used to reduce interaction errors [2]. Exploiting symmetry, discs are not magnetically attracted and cannot misalign due to geometry, in comparison to the assembly errors of the seed components by Hosokawa et al. The geometry of these components prevents incompatible substructures from forming.

Sector and disc-ray component sets use identical circular tray environments (similar to petri dishes). The dimensions of a tray include an inner wall diameter of 118.55 mm and height of 6 mm. A tray is fastened horizontally to an orbital

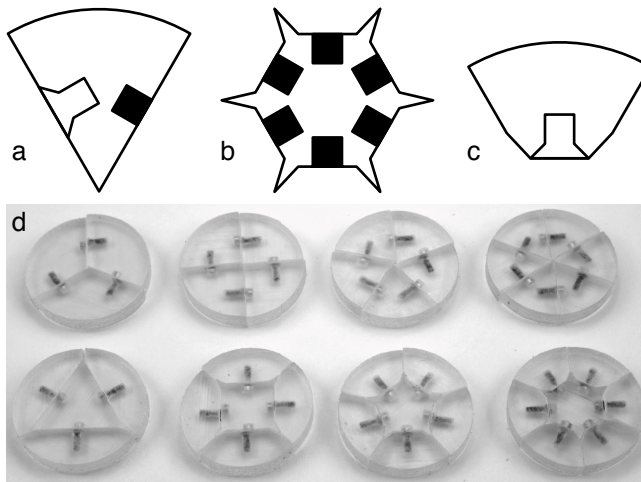


Fig. 2. Abstract examples of C_6^S (a), C_6^D (b), and C_6^R (c), and the 8 types of target structure with 3D printed components embedded with magnets and micro-screws (d)

shaker, providing vibrational energy to the passive, mechanical components; a lid constrains the components to the tray surface. Each tray accommodates 10 target structures. The shaking speed and the tray dimensions are based on preliminary experiments conducted by the authors. Complete physical specifications of the tray, and the sector, disc, and ray components are provided in [3].

4 Experiment

The benefit of our geometrical approach will be shown by a statistically significant difference in yield between the sector and disc-ray component sets. The null hypothesis, H_0 , is that there is no difference in the yield of self-assembled circular target structures between the sector and disc-ray component sets. The alternative hypothesis, H_1 , is that there is a difference in the yield of self-assembled circular target structures between the sector and disc-ray component sets.

To test H_0 , 10 trials of this experiment are conducted. The dependent variable is the number of self-assembled target structures. The independent variable is the set of components. A trial is conducted for C_{3-6}^S , and C_{3-6}^D and C_{3-6}^R . Enough components are provided to self-assemble 10 target structures. The control group is the set of sector components and the experimental group is the set of disc-ray components (subgroups denoted by CG_y^S and EG_y^{DR} , where $y \in \{3, 4, 5, 6\}$).

The structure of all the components is made from a photopolymer resin, fabricated using an Objet Eden 3D printer. Finishing the components includes manually inserting permanent magnets and micro-screws. Four trays are used in the experiment, in order to conduct trials in parallel. The base of each tray is made from ABS plastic, fabricated using a Stratasys Fortus 3D printer. The laser-cut tray lids are made from clear acrylic. A full list of the materials and procedures for constructing the components and trays is provided in [3].

The four trays are fastened to an Excella E5 orbital shaker. At the start of each trial, all the components in a subgroup are randomly placed into one of the four trays (ensuring components are not initially aligned to self-assemble). The shaker is turned on and set to 300 rpm. The shaker is turned off after 300 seconds. A complete description of the experiment procedure is provided in [3]. At the end of each trial, the following four quantitative measurements are recorded: (1) the number of self-assembled target structures, (2) the number and type of compatible substructures, (3) the number and type of incompatible substructures, and (4) the number and type of errors (misalignment and defects).

5 Results

Frequency histograms for the number of self-assembled target structures, for the control and experimental groups, are provided in Fig. 3. The frequency histograms show that the yield in target structures is not normally distributed for both groups. Therefore, the non-parametric Mann-Whitney-U test is used to test H_0 [6]. The sample size, and median and mean number of self-assembled target structures for the control group are 40, 7, and 6, for the experimental

group are 40, 9, and 9. The calculated critical value, U , is 1,394. The difference in yield between the control and experimental groups is statistically significant (two-tailed test, p -value < 0.001). Therefore, we reject H_0 .

A deeper investigation illustrates the difference in yield and waste between the subgroups, CG_{3-6}^S and EG_{3-6}^{DR} . Here, waste is considered to be any substructure that remains at the end of a trial. The first box-and-whisker plot shows the interquartile range and the minimum and maximum number of self-assembled target structures, yield, for each subgroup (Fig. 3). The second box-and-whisker plot shows the interquartile range and the minimum and maximum number of compatible substructures and incompatible substructures for CG_{3-6}^S (Fig. 3). The third box-and-whisker plot shows the interquartile range and the minimum and maximum number of disc-based substructures and the number of single ray components for EG_{3-6}^{DR} (Fig. 3).

Notably, CG_4^S achieved a consistently high yield, and only had incompatible substructures occur in trials 9 and 10 (substructures with 3, 3, and 2 components for both trials). This high yield might be attributed to symmetry, but requires further investigation. Trials 1 and 2 of CG_5^S did not have incompatible substructures, and instead had compatible substructures with 3 and 2 components for both trials. For CG_5^S , trials 3 to 10 did not have any compatible substructures. Only trials 2, 3, and 4 of CG_6^S had compatible substructures. The typical number of components in the incompatible substructures for CG_5^S and CG_6^S were one less than required for the target structures. For trials 1–10, for EG_{3-5}^{DR} , only a single, unattached C_{3-5}^R prevented 10 target structures from self-assembling. For EG_6^{DR} , trials 1, 5, and 6 had a single C_6^R each. Trial 3 had one C_6^D with one empty slot and a second C_6^D with two empty slots (three C_6^R). Trials 4, 7, and 9 had two C_6^D with one empty slot each (two C_6^R). And, trial 8 had one C_6^D with two empty slots (two C_6^R). A larger sample size is required to directly compare CG_{3-6}^S to EG_{3-6}^{DR} . Fig. 4 shows examples from the experiment.

No misalignment errors were observed. Only one defect error occurred in trial 4 of EG_6^{DR} , where a magnet from C_6^D dislodged and self-assembled two C_6^R . Despite this error, the difference in yield between the sector and disc-ray groups is statistically significant. Even when increasing the number of components in a target structure, we attribute this difference to our geometrical approach.

6 Future Work

We are currently investigating three extensions to our approach. First, we are setting-up a camera tracking system to record the physical trials, in order to conduct deeper analysis into the self-assembly process of target structures in parallel. One of the shortcomings of the geometrical approach described here is that it is restricted to a single layer of components. Second, we are investigating new geometries in order to scale the number of components to additional layers, and leverage the new geometries to autonomously stage the construction of each layer from the inside-out. For example, more complex arrangements of multiple permanent magnets can be used to selectively attract and repel a wider variety

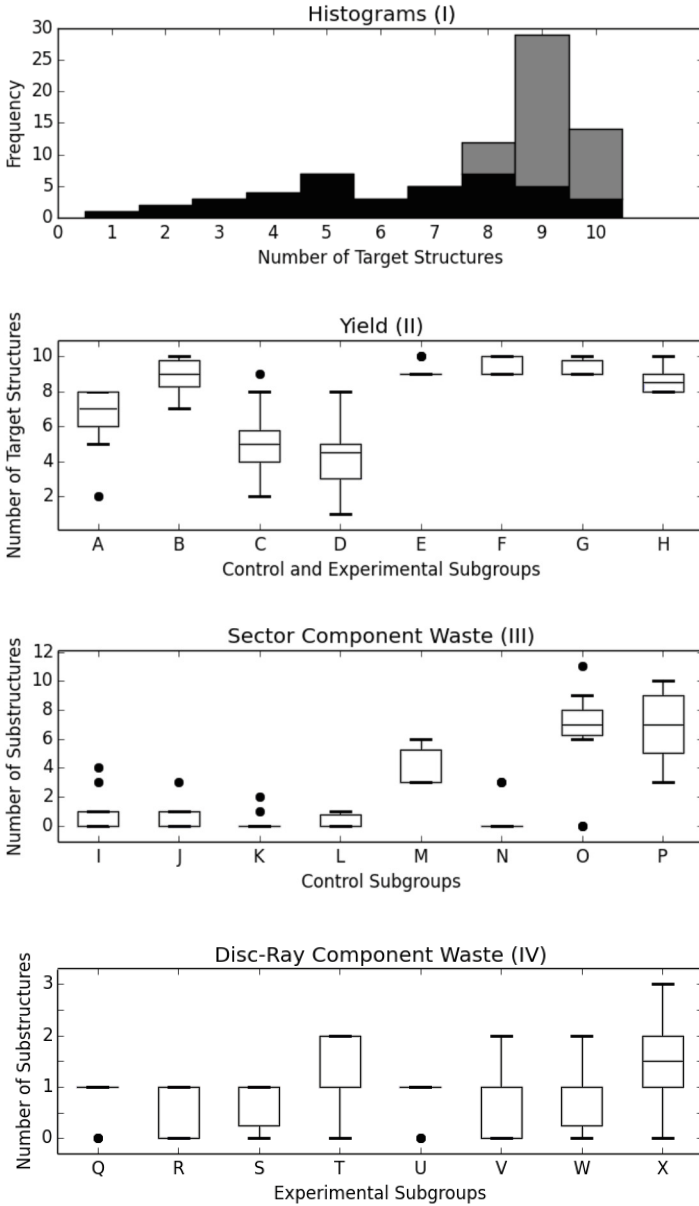


Fig. 3. (I) Histograms for control (black) and experimental (grey) groups. (II) Box-and-whisker plot for the number of self-assembled target structures in 10 trials for CG_{3-6}^S (A-D) and EG_{3-6}^{DR} (E-H). (III) Box-and-whisker plot for the number of compatible, CG_{3-6}^S (I-L), and the number of incompatible, CG_{3-6}^S (M-P), substructures in 10 trials. (IV) Box-and-whisker plot for the number of disc-based substructures, EG_{3-6}^{DR} (Q-T), and the number of single ray components, EG_{3-6}^{DR} (U-X), in 10 trials.

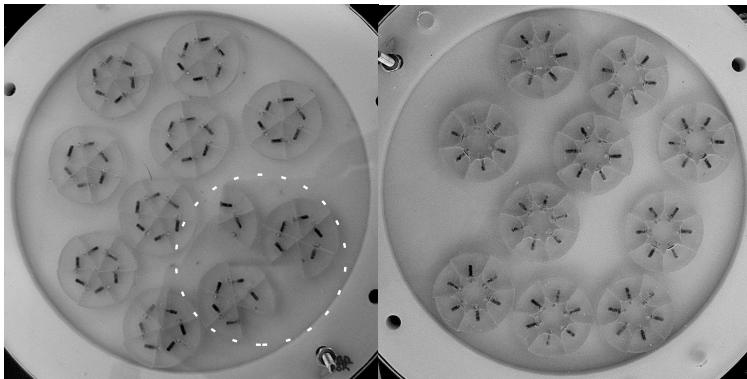


Fig. 4. Example of incompatible substructures (2, 5, and 5 component substructures, CG_6^S , trial 10; left), and of self-assembled target structures (EG_6^{DR} , trial 10; right)

of component types (i.e. components in different layers), while self-assembling structures from the inside-out [2]. Third, we are considering new component and environment designs in order for the components to move and self-assemble in three spatial dimensions.

7 Conclusions

In this work, we presented our geometrical approach to the incompatible substructure problem. The primary contribution of this work is that our approach eliminates incompatible substructures from forming during the self-assembly process. The secondary contributions of this work includes the reduction of component interactions errors by combining a permanent-ferromagnetic binding mechanism with the key-lock principle, and a physical experiment that produced a high yield in target structures (even as the number of components increased) while reducing waste. The evidence from this experiment supports our geometrical approach to the incompatible substructure problem in parallel self-assembly.

Acknowledgments. This work has been partially supported by a Postdoctoral Fellowship provided by the Natural Sciences and Engineering Research Council of Canada, and by the European Research Council through the ERC Advanced Grant “E-SWARM: Engineering Swarm Intelligence Systems” (contract 246939). Marco Dorigo acknowledges support from the Belgian F.R.S.–FNRS. We thank the Direct Manufacturing Research Center and Hamburg University of Applied Sciences for 3D printing the parts used in the experiment.

References

1. Barish, R.D., Schulman, R., Rothmund, P.W.K., Winfree, E.: An information-bearing seed for nucleating algorithmic self-assembly. *Proc. Natl. Acad. Sci. U.S.A.* 106(15), 6054–6059 (2009)

2. Bhalla, N., Bentley, P.J., Vize, P.D., Jacob, C.: Staging the self-assembly process: Inspiration from biological development. *Artificial Life* 20(1), 29–53 (2014)
3. Bhalla, N., Ipparhi, D., Klemp, E., Dorigo, M.: A geometrical approach to the incompatible substructure problem in parallel self-assembly: supplementary material. Tech. Rep. TR/IRIDIA/2014-010, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2014)
4. Dagliyan, O., Shirvanyants, D., Karginov, A.V., Dinga, F., Feea, L., Chandrasekarana, S.N., Freisingerd, C.M., Smolend, G.A., Huttenlocherd, A., Hahnc, K.M., Dokholyana, N.V.: Rational design of a ligand-controlled protein conformational switch. *Proc. Natl. Acad. Sci. U.S.A.* 110(17), 6800–6804 (2013)
5. Gautam, V.K., Haddow, P.C., Kuiper, M.: Reliable self-assembly by self-triggered activation of enveloped DNA tiles. In: Dediu, A.-H., Martín-Vide, C., Truthe, B., Vega-Rodríguez, M.A. (eds.) *TPNC 2013. LNCS*, vol. 8273, pp. 68–79. Springer, Heidelberg (2013)
6. Hettmansperger, T.P., McKean, J.W.: *Robust nonparametric statistical methods*. Chapman & Hall/CRC Press, Boca Rotan (2010)
7. Hosokawa, K., Shimoyama, I., Miura, H.: Dynamics of self-assembling systems: Analogy with chemical kinetics. *Artificial Life* 1(4), 413–427 (1994)
8. Klavins, E.: Programmable self-assembly. *IEEE Control Systems Magazine* 27(4), 43–56 (2007)
9. Mastrangeli, M., Abbasi, S., Van Hoof, C., Celis, J.P., Böhringer, K.F.: Self-assembly from milli- to nanoscales: methods and applications. *Journal of Micromechanics and Microengineering* 19(8), 1–37 (2009)
10. Mendes, A.C., Baran, E.T., Reis, R.L., Azevedo, H.S.: Self-assembly in nature: using the principles of nature to create complex nanobiomaterials. *WIREs Nanomedicine and Nanobiotechnology* 5(6), 582–612 (2013)
11. Miyashita, S., Nagy, Z., Nelson, B.J., Pfeifer, R.: The influence of shape on parallel self-assembly. *Entropy* 11(4), 643–666 (2009)
12. Pelesko, J.A.: *Self Assembly: The Science of Things that Put Themselves Together*. Chapman & Hall/CRC Press, Boca Rotan (2007)
13. Saitou, K.: Conformational switching in self-assembling mechanical systems. *IEEE Transactions on Robotics and Automation* 15(3), 510–520 (1999)
14. Schneiter, A.A., Miller, J.F.: Description of sunflower growth stages. *Crop Science* 21(6), 901–903 (1981)
15. Whitesides, G.M., Boncheva, M.: Beyond molecules: Self-assembly of mesoscopic and macroscopic components. *Proc. Natl. Acad. Sci. U.S.A.* 99(8), 4769–4774 (2002)
16. Whitesides, G.M., Grzybowski, B.: Self-assembly at all scales. *Science* 295(5564), 2418–2421 (2002)
17. Winfree, E., Liu, F., Wenzler, L.A., Seeman, N.C.: Design and self-assembly of two-dimensional DNA crystals. *Nature* 394, 539–544 (1998)

Application of Evolutionary Methods to Semiconductor Double-Chirped Mirrors Design

Rafał Biedrzycki¹, Jarosław Arabas¹, Agata Jasik², Michał Szymański²,
Paweł Wnuk³, Piotr Wasylczyk³, and Anna Wójcik-Jedlińska²

¹ Institute of Electronic Systems, Warsaw University of Technology, Poland
{rbiedrzy, jarabas}@elka.pw.edu.pl

² Institute of Electron Technology, Warsaw, Poland
{ajasik, mszyman, awojcik}@ite.waw.pl

³ Faculty of Physics, Institute of Experimental Physics, University of Warsaw, Poland
{Pawel.Wnuk, Piotr.Wasylczyk}@fuw.edu.pl

Abstract. This paper reports on a successful application of evolutionary computation techniques to the computer aided design of a dedicated highly dispersive mirror which is used in an ultrafast laser. The mirror is a GaAs plate coated with many interleaving layers of GaAs/AlAs and SiO₂/Si₃N₄ layers whose thickness undergo optimization. We report and compare results obtained by leading global optimization techniques: Covariance Matrix Adaptation Evolution Strategy and Differential Evolution, as well as few efficient local optimization methods: Nelder-Mead and variable metric. The evolutionary designed mirror has been manufactured by the Molecular Beam Epitaxy technology and the measurements confirmed successful implementation of the instrument.

Keywords: CMA-ES, Differential Evolution, Double Chirped Mirror.

1 Introduction

In the evolutionary computation history, it has been relatively early observed that the driving force of development of methods is their application for global optimization. Since the 1990ies, many benchmark problems have been adopted from the global optimization domain and have been used to test efficiency of evolutionary computing techniques. In 2005 and 2009, two families of benchmark sets were born: CEC [1] (starting from CEC2005) and BBOB [2] (starting from BBOB 2009). These benchmark sets include not only the test problems, but also specify procedures to test optimization methods and to interpret their results. With the use of benchmark sets it is possible to compare two optimization methods to answer the question which of them will, on average, provide better results than the other. This allows to organize competitions to select algorithms that perform most efficiently on the benchmark set.

A practitioner's perspective is somewhat different, since he/she is not interested in the long-lasting tuning of many parameters of a beloved algorithm to make it perform better than other competitors. Instead, the practitioner is focused on obtaining an acceptable solution of a particular problem with the least

effort. Results of benchmarking may be useful to perform preliminary selection of optimization methods. Parameter-free methods are obviously preferred over these with many settings to be tuned, since the practitioner is interested in easy use rather than in the development of optimization methods. Hence, software packages with predefined parameter setting will usually be preferred.

In the presented work we followed this practitioner's point of view. We report results of an application of two packages for the R environment [3], `DEoptim` [4] and `cmaes` [5], which implement DE [6] and CMA-ES [7] — two global optimization methods from the evolutionary family which have been usually on top of efficiency rankings based on benchmarking [1,8]. These methods have a relatively small number of user-defined parameters and the aforementioned packages provide default parameter settings. We applied these methods to the computer aided design of a mirror, which was dedicated for an ultrafast laser. The mirror design was represented as a vector of 126 real numbers. We have performed multiple runs of compared methods and we report statistics of the results to select the optimization method. Further tests were carried out to assess influence of starting point generation method upon mirror quality. The best structure of the mirror, which was found by the CMA-ES, was performed in a semiconductor technology and successfully tested for conformance with the design assumptions. This structure, after antireflective layer deposition, will be used to build the ultrafast laser in the near future.

2 Designing Mirrors for Ultrafast Lasers

Ultrafast lasers, which generate optical pulses in the picosecond and femtosecond range, have found numerous applications, e.g., in industry [9], biology and medicine [10]. The basic technique allowing for generation of ultrashort pulses is modelocking. We distinguish between active and passive modelocking, while the latter provides much shorter pulses [11]. The underlying principle of this technique is to induce a fixed phase relationship between the modes inside the resonant cavity. In order to achieve it, all kinds of dispersion appearing in the cavity must be compensated. This can be achieved either by using the prism pair or grating pair, or a multilayer mirror, called also a Chirped Mirror (CM) [12].

The CM is a structure that contains an alternating sequence of layers of two different optical materials. Thickness of each pair of layers is linearly variable. In effect, light from a range of wave length is effectively reflected, and longer wavelengths penetrate the structure deeper than shorter ones. Thus the delay of the reflected light depends on its wavelength. This process can be characterized by the Group Delay Dispersion (GDD) coefficient and typically, the dependence of GDD on the wavelength is very irregular, which is undesirable. This effect can be reduced in a Double Chirped Mirror (DCM) where thickness of layers is changing non-linearly. Properly designed DCM should exhibit high reflection in a broad wavelength band together with non-oscillating dependence of GDD on the wavelength.

There are several theoretical approaches to design DCMs in an analytical way. Coupled mode theory is a perturbational approach for analyzing the coupling of

vibrational systems (inter alia optical) in space or in time [13]. WKB approximation is a method for finding approximate solutions to linear partial differential equations with spatially varying coefficients [14]. For example these methods have been used by Matuschek to derive formulas for reflectivity and group delay [12]. Such analytical works undoubtedly enabled deeper understanding of chirped mirrors, but the GDD characteristics of the resulting DCM exhibited oscillatory behavior and it needed further fine tuning.

The majority of approaches to designing DCMs have used the transfer matrix method [15] to model the light reflection from multilayer structure and to optimize layers' thickness to achieve desired DCM properties. Such approach was used by Yakovlev et al. [16] who applied a memetic algorithm to design a 49-layer dielectric $\text{SiO}_2/\text{TiO}_2/\text{Ta}_2\text{O}_5$ chirped mirror. Dielectric materials were also used by other authors, e.g. Yan-Zhi et al. [17] applied a needle optimization technique [18] to design 52-layers structure with $\text{GDD} \approx -60 \text{ fs}^2$. They have also reported successful physical implementation of the design. Another successful DCM implementation has been reported by Pervak et al. [19] who used a commercial package `OptiLayer` to design a dielectric DCM for $\text{GDD} \approx -2500 \text{ fs}^2$. Unfortunately, all aforementioned articles did not report important information about optimization methods, like the objective function formulation, settings of parameters or the starting point, which does not allow to reproduce the results.

This paper briefly describes the design process of a Semiconductor DCM (SDCM). Semiconductor materials, GaAs and AlAs, which are used for the mirror realization, have high refractive indexes (3.51 and 2.95) which makes it hard to avoid reflection of the light from the mirror surface. For this reason, the SDCM should be additionally covered by an Anti-Reflective (AR) layer composed of few layers of dielectric materials. The difference of refractive indexes of GaAs and AlAs is lower than in the case of dielectric materials used by other authors, therefore more layers are needed to achieve mirror parameters comparable to dielectric DCMs. More layers means also more interfaces between layers which increases risk of harmful interferences that disrupt the GDD curve. DCMs with larger number of layers are also harder to optimize due to the "curse of dimensionality".

The SDCM production process consists of two technological stages: the Molecular Beam Epitaxy (MBE, see Fig. 1) to produce semiconductor layers on a two-inch GaAs substrate and the plasma-enhanced chemical vapor deposition technology to coat it with dielectric AR layers.

3 SDCM Design Problem Formulation

SDCM Model. Mathematical model of the reflection process assumes that a time-harmonic plane wave falls on the stratified medium, which is schematically depicted in Fig. 2. The electric field $E(x)$ satisfies the Helmholtz equation [13]:

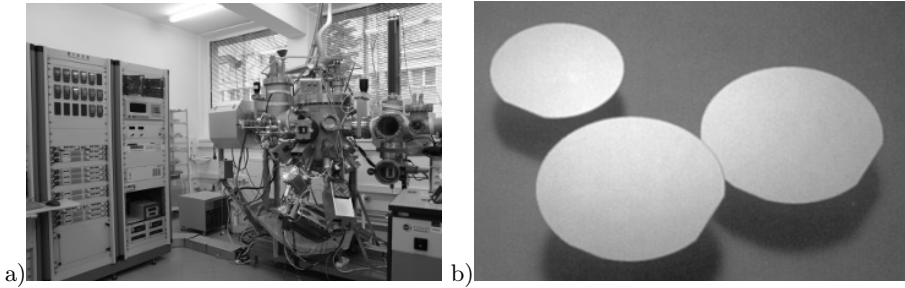


Fig. 1. a) MBE reactor which is used to produce SDCM, b) 2” substrate wafers on which the mirror is deposited

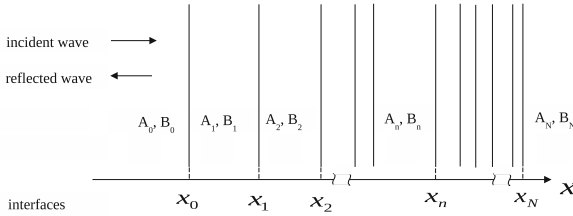


Fig. 2. Schematic view of a stratified medium

$$\frac{d^2 E(x)}{dx^2} + [(\tilde{n}_n k_0)^2 - \beta^2] E(x) = 0, \tag{1}$$

where \tilde{n}_n is the complex refractive index of the n -th layer, $\beta = n_{\text{eff}} k_0$ is the propagation constant, $k_0 = 2\pi/\lambda_0$, λ_0 is the free space wavelength and n_{eff} is the effective refractive index. In the n -th layer, the field $E(x)$ is a superposition of waves traveling towards the left and the right direction:

$$E(x) = A_n \exp(iK_n x) + B_n \exp(-iK_n x), \tag{2}$$

where $K_n = k_0 \sqrt{\tilde{n}_n^2 - n_{\text{eff}}^2}$, and A_n, B_n are constants. Assuming $A_0 = 1, B_{N+1} = 0$, we identify B_0 and A_{N+1} , as coefficients of mirror reflectivity and mirror transmission, respectively.

DCM analysis method based on the transfer matrix [15] is facilitated on the fact that the field and its first derivative must be continuous at the interfaces. According to this method we derive a matrix equation of the form:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} B_0 \\ A_{N+1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \tag{3}$$

Elements a_{kl} and b_k depend on thickness and refractive indexes of layers. Solution of (3) yields the key parameters of SDCM:

$$R = |B_0|^2, \text{GDD} = \frac{d^2(\arg B_0)}{d\omega^2}. \tag{4}$$

where $\omega = 2\pi c/\lambda_0$ denotes the angular frequency, c is the light velocity, R is the reflectivity coefficient, and GDD is the Group Delay Dispersion.

Representation of Solutions and Constraints. On the basis of our experimentally verified knowledge it was assumed that SDCM should consist of 120 semiconductor layers and 6 dielectric AR layers. The structure is represented as a sequence of real numbers which define thickness of layers. Layers are numbered according to the sequence in which they are penetrated by the light which enters into the mirror structure. Hence, the sequence starts from AR layers, i.e. from 3 pairs of SiO₂ and TiO₂. They are followed by 60 pairs of GaAs/AlAs. Upper limit of each semiconductor layer thickness was assumed to 0.2 μm. Lower limits of thickness were 7 nm for dielectric layers and 0.6 nm for semiconductor layers, except for the the first semiconductor layer which had to be at least 30 nm thick. The first semiconductor layer is thicker to prevent the destructive oxidation of AlAs.

Objective Function. The design objective was to achieve the following mirror parameters: GDD = -2500 ± 100 fs², R ≥ 0.999 in a band from λ_l = 1.02 μm to λ_h = 1.04 μm. Having performed a series of trials we decided to use the following objective function to be minimized:

$$q = a \cdot q_R + (1 - a) \cdot q_D \tag{5}$$

where q_D and q_R are the distance from the target levels of GDD and R, and a is scalarization coefficient. It is known that optimization of GDD is much harder than R, therefore a was set to 0.01. Values of q_D and q_R are defined as

$$q_D = b \sum_{\lambda} (D_0 - D_{\lambda})^2, q_R = u \sum_{\lambda} (1 - R_{\lambda})^2 \tag{6}$$

where b, u are the scaling factors responsible for bounding values of q_D and q_R in range (0, 1), λ is a wavelength sampled from range (λ_l, λ_h), D₀ is the required GDD value, D_λ and R_λ are the GDD and R values calculated for wavelength λ. Note that the objective function values come from the range (0, 1). The objective function is very irregular — sample two-dimensional cross-sections are provided in Fig. 3.

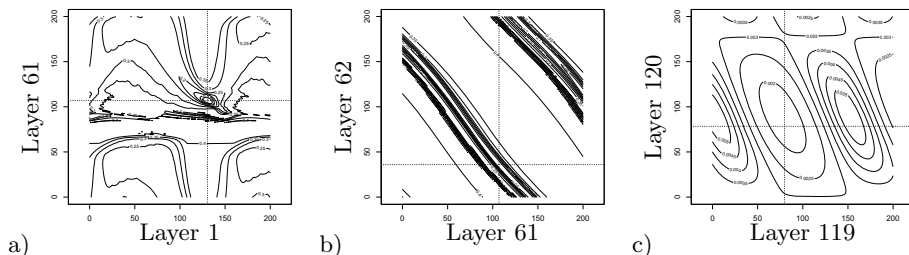


Fig. 3. Fitness function contour plot for varying thickness of: a) layers 1 and 61, b) layers 61 and 62, c) layers 119 and 120. Dotted lines marks values achieved after mirror optimization

4 Optimization of the SCDM Design

The solution of the SCDM design problem was a two stage process. In the first stage we performed experimental comparison of results obtained by several optimization methods. For all compared methods it was observed that the quality of results was unacceptable when the initial SCDM structure was set randomly with uniform distribution in the admissible area. Acceptable results were obtained when the starting structure was the design obtained by application of the theory provided by Matuschek [12]. This starting point, which will be denoted by DCM_0 , was used for selection of the best performing optimization method. In the second stage, for the selected method, different starting points were examined which finally gave the setup that would allow for obtaining high quality results. In addition to DCM_0 , we considered the Bragg mirror, which was designed to achieve high reflectivity for a single wavelength, and the single chirped Bragg mirror for which high-reflectivity bandwidth is significantly increased. Plots of the layers' thickness for these starting points are shown in Fig. 4.

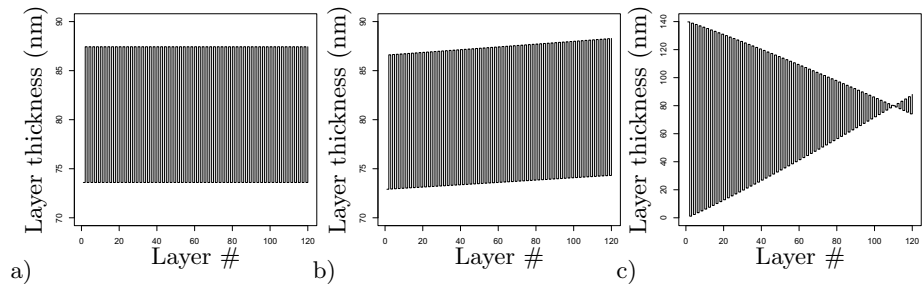


Fig. 4. Layers thickness of: a) Bragg mirror, b) single chirped Bragg mirror, c) DCM_0

Our previous experience with similar design problems [20,21] showed that metaheuristics from the evolutionary family would be good candidates for solving engineering problems. According to results of competitions [8,22] we selected Differential Evolution (DE) [6] and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [7]. We also tested two efficient local optimization methods: nonlinear simplex by Nelder and Mead (NM), and the L-BFGS-B algorithm. The whole process of simulation and optimization of SCDM structures, and of testing optimization methods, was performed under the R environment [3]. We used R packages: `cmaes` for the CMAES [5] and `DEoptim` for the DE [4]. Implementations of NM was provided by Bihorel et al. [23] and L-BFGS-B [24] was taken from the `optim` package.

Selection of the Most Efficient Optimization Method. In preliminary tests we observed that DE was able to improve the quality of the initial design using default parameter settings but the GDD spectral characteristics was far

from satisfactory. Default parameter settings for CMA-ES yielded no improvement to the initial design. Therefore we performed manual parameter tuning and we ended up with different settings which gave acceptable solutions. In the case of CMA-ES, Hansen suggested [25] that in most cases default values of parameters which are used by the "reference" implementation of CMA-ES should be appropriate if design variables are scaled to fit into the range $(0, 10)$. Our design variables are bounded by 0.0006 and 0.2, so they are not in CMA-ES "favorite" range. Instead of scaling, we changed the σ parameter and we observed good results for $\sigma = 0.0003$. We used the default DE version in the `DEoptim` package, which was the DE/local-to-best/1/bin algorithm. The best results were obtained for CR=1 and the remaining parameters were set to the default values. The DE method must be started with a diversified population. To achieve this, the initial population was filled with Gaussian perturbations of the starting point.

An additional comments needs to be made on the NM method. We tested its three different implementations: NM₁ [26], NM₂ [23] and NM₃ from the `optim` package. These three versions gave substantially different results: NM₁ was 10 times worse than NM₃ and 33 times worse than NM₂. For this reason we report only the results by NM₂.

In Fig. 5a, box and whisker plots are provided of the distribution of the best objective function values obtained in 25 independent runs of compared algorithms; for NM and L-BFGS-B a single value is provided, since they are deterministic methods and the starting point is always the same.

According to the results, both DE and CMA-ES outperformed their competitors when starting from the initial design DCM₀. CMA-ES revealed greater variability than DE, but in the same time median of its results was significantly better than for DE. For this reason, an additional experiment was made to check if the CMA-ES efficiency could be further improved by using a different starting point.

Sensitivity to the Starting Point. From preliminary experiments we learned that the CMA-ES was unable to generate acceptable solutions when started from a random solution. The DCM₀ structure is a nearly optimal design which introduces a risk that it would become a trap which would not allow CMA-ES to generate better solutions which are substantially different than the initial structure. In Fig. 5b we provide statistics of objective function values achieved for 25 independent runs of CMA-ES for three starting points depicted in Fig. 4. The DCM₀ structure appeared the worst among all considered starting points, although this structure is the result of the most advanced analytical approach to the DCM design. Much better results were obtained for simple Bragg and linearly chirped mirrors — in most of runs CMA-ES was able to approach the global minimum of the fitness. For the single chirped structure the number of successful runs was greater so we conclude that this was the best choice of starting point for CMA-ES.

Solutions generated from this starting point revealed another advantage over those generated from DCM₀. According to the technology experts, they were easier to produce, because thickness of neighboring layers is less diversified. Another advantage of obtained designs was that they allowed for reduction of the

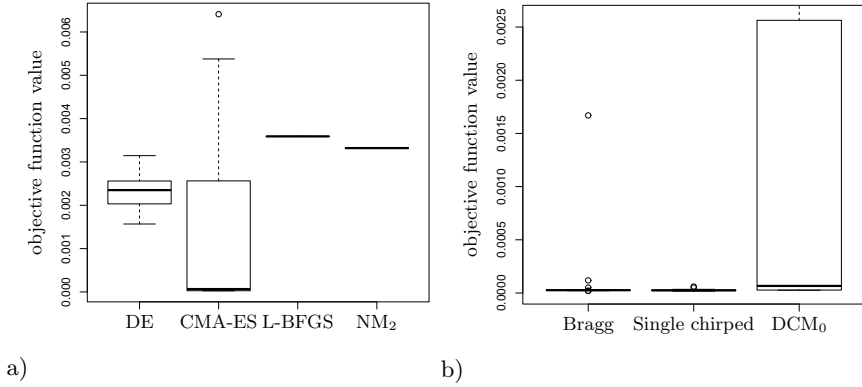


Fig. 5. Box and whisker plots of the best objective function values obtained in 25 independent runs of each method; a) Results for various optimization methods with starting point DCM₀, b) results of CMA-ES for three different starting points

number of layers. We took the best result obtained in all experiments and we removed 10 semiconductor layers and 4 dielectric layers (in both cases layers were removed from the base side). Additionally we changed TiO₂ dielectric into Si₃N₄. This was the starting point for CMA-ES which yielded another solution that satisfied design criteria. This solution was the final design (see Fig. 6a).

5 Implementation and Tests of the Designed Mirror

The final design obtained by CMA-ES was used to produce the semiconductor part of the mirror in the MBE reactor (Fig. 1) in the Institute of Electron Technology (ITE). Prior to coating the mirror with AR layers, its R and GDD factors were measured by the ITE and Institute of Experimental Physics in the assumed wavelength band.

Measurements of GDD were performed using a modified Michelson interferometer illuminated with a white light source, delivered by a halogen bulb, to obtain GDD characteristic in a broad range of wavelengths. A reference curve of GDD vs. wavelength has been generated from the SDCM model and it was verified if it is approximately matched by the measured curve. The results, which are presented in Fig. 6b, indicate a good agreement of both curves.

Mirror reflectance was measured using a dark configuration of a standard optical set-up, where a broadband source light is dispersed by a monochromator and formed to a quasi-parallel probe beam which illuminates the sample surface at normal incidence. Reflected light is optically collected and focused onto a silicon detector. The measurement confirmed that the mirror reflectance matched the design criteria.

The measurements confirm that the mirror was correctly produced by the MBE process, and it therefore can undergo the second phase of production –

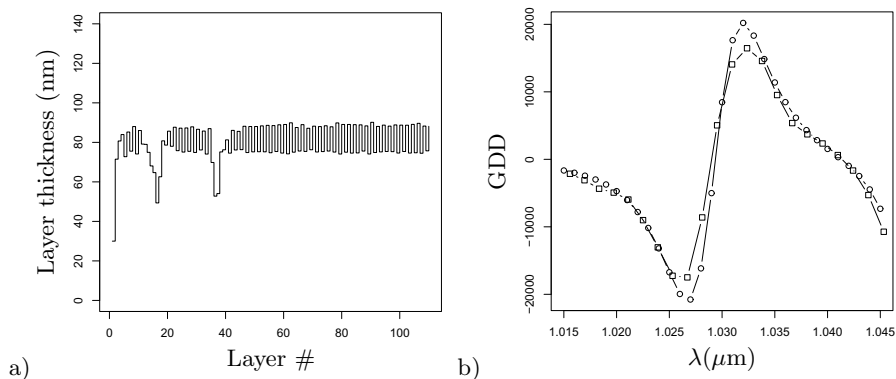


Fig. 6. a) Layers thickness of designed SDCM. b) Comparison of theoretical (circles) and measured (squares) GDD characteristics.

deposition of 2 dielectric layers of AR coating. The resulting product will be used as a component of an ultrafast laser constructed by Institute of Experimental Physics.

6 Conclusions

We presented a successful practical application of evolutionary techniques to the design of a Double Chirped Mirror which was produced in the semiconductor technology. Application of CMA-ES yielded a solution that outperformed engineering practice of designing DCM structures. Obtained design was also acceptable from the production technology point of view. The mirror has been performed and it will be used to build a compact ultrafast laser.

The design problem size exceeds an informal limit of 100 dimensions which was suggested for CMA-ES applications. Quality of obtained results is sensitive to the starting point which indicates that analyzed methods realize a form of a “globalized local search” — they are only partially robust against getting trapped into local optima. The objective function used for the design, which has been implemented in the R language, is very irregular and it may serve as a benchmark problem for testing global optimization methods.

References

1. Liang, J., et al.: Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Technical report, Comp. Intell. Lab., Zhengzhou University and Nanyang Technological University (2013)
2. INRIA: Black-box optimization benchmarking (BBOB) (2013), <http://coco.gforge.inria.fr/doku.php?id=bbob-2013>
3. R Core Team: R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2013)

4. Ardia, D., et al.: DEoptim: Differential Evolution in R (2013), Package ver. 2.2-2
5. Trautmann, H., Mersmann, O., Arnu, D.: cmaes: Covariance Matrix Adapting Evolutionary Strategy (2011) R package version 1.0-11
6. Storn, R.: Differential evolution research – trends and open questions. In: Chakraborty, U. (ed.) *Advances in Differential Evolution*. SCI, vol. 143, pp. 1–31. Springer, Heidelberg (2008)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* 9(2), 159–195 (2001)
8. Hansen, N., et al.: Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: GECCO, pp. 1689–1696. ACM (2010)
9. Weiler, S.: Ultrafast lasers-high-power pico-and femtosecond lasers enable new applications. *Laser Focus World* 47(10), 55 (2011)
10. Braun, M., Gilch, P., Zinth, W.: *Ultrashort laser pulses in biology and medicine*. Springer (2008)
11. Keller, U.: Recent developments in compact ultrafast lasers. *Nature* 424(6950), 831–838 (2003)
12. Matuschek, N.: *Theory and design of double-chirped mirrors*. PhD thesis, ETH Zürich (1999)
13. Marcuse, D.: *Theory of dielectric optical waveguides*. Academic, New York (1974)
14. Bialynicki-Birula, I., Cieplak, M., Kaminski, J.: *Theory of quanta*. Oxford University Press, New York (1992)
15. Chilwell, J., Hodgkinson, I.: Thin-films field-transfer matrix theory of planar multilayer waveguides and reflection from prism-loaded waveguides. *JOSA A* 1(7), 742–753 (1984)
16. Yakovlev, V., Tempea, G.: Optimization of chirped mirrors. *Applied Optics* 41(30), 6514–6520 (2002)
17. Yan-Zhi, W., et al.: Design and fabrication of chirped mirror. *Chinese Physics Letters* 26(9) (2009)
18. Tikhonravov, A.V.: Needle optimization technique: the history and the future. In: *Optical Science, Engineering and Instrumentation 1997*, International Society for Optics and Photonics, pp. 2–7 (1997)
19. Pervak, V., et al.: High-dispersive mirrors for femtosecond lasers. *Optics Express* 16(14), 10220–10233 (2008)
20. Adamski, K.: Evolutionary algorithm versus variable metric method in digital FIR filter design. In: *EUROCON 2007*, pp. 116–121 (2007)
21. Arabas, J., Miazga, P.: Computer aided design of a layout of planar circuits by means of evolutionary algorithms. *CIT. Journal of Computing and Information Technology* 7(1), 61–76 (1999)
22. Rios, L., Sahinidis, N.: Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization* 56(3), 1247–1293 (2013)
23. Bihorel, S., Baudin, M.: Neldermead: R port of the Scilab neldermead module (2014), R package version 1.0-8
24. Zhu, C., et al.: Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* 23(4), 550–560 (1997)
25. Hansen, N.: CMA-ES source code (2014), https://www.lri.fr/~hansen/cmaes_inmatlab.html
26. Varadhan, R., Borchers, H.W.: dfoptim: derivative-free optimization (2011), R package version 2011.8-1

Evolving Neural Network Weights for Time-Series Prediction of General Aviation Flight Data

Travis Desell¹, Sophie Clachar¹, James Higgins², and Brandon Wild²

¹ Department of Computer Science, University of North Dakota, USA
tdesell@cs.und.edu, sophine.clachar@my.und.edu

² Department of Aviation, University of North Dakota, USA
{jiggins,bwild}@aero.und.edu

Abstract. This work provides an extensive analysis of flight parameter estimation using various neural networks trained by differential evolution, consisting of 12,000 parallel optimization runs. The neural networks were trained on data recorded during student flights stored in the National General Aviation Flight Database (NGAFID), and as such consist of noisy, realistic general aviation flight data. Our results show that while backpropagation via gradient and conjugate gradient descent is insufficient to train the neural networks, differential evolution can provide strong predictors of certain flight parameters (10% over a baseline prediction for airspeed and 70% for altitude), given the four input parameters of airspeed, altitude, pitch and roll. Mean average error ranged between 0.08% for altitude to 2% for roll. Cross validation of the best neural networks indicate that the trained neural networks have predictive power. Further, they have potential to act as overall descriptors of the flights and can potentially be used to detect anomalous flights, even determining which flight parameters are causing the anomaly. These initial results provide a step towards providing effective predictors of general aviation flight behavior, which can be used to develop warning and predictive maintenance systems, reducing accident rates and saving lives.

Keywords: Time-Series Prediction, Asynchronous Differential Evolution, Neural networks, Flight Prediction, Aviation Informatics.

1 Motivation

General aviation comprises 63% of all civil aviation activity in the United States; covering operation of all non-scheduled and non-military aircraft [12,24]. While general aviation is a valuable and lucrative industry, it has the highest accident rates within civil aviation [21]. For many years, the general aviation accident and fatality rates have hovered around 7 and 1.3 per 100,000 flight hours, respectively [1]. The general aviation community and its aircraft are very diverse, limiting the utility of the traditional flight data monitoring (FDM) approach used by commercial airlines.

The National General Aviation Flight Information Database (NGAFID) has been developed at the University of North Dakota as a central repository for general aviation flight data. It consists of per-second flight data recorder (FDR) data from three fleets of aircraft. As of January 2014, the database stores FDR readings from over 208,000

flights, with more being added daily. It currently stores almost 400 million per-second records of flight data. The NGAFID provides an invaluable source of information about general aviation flights, as most of these flights are from aviation students, where there is a wider variance in flight parameters than what may normally be expected within data from professionally piloted flights.

This research presents initial work done using data from the NGAFID for the prediction of flight data parameters. Five flights were selected from the NGAFID, and a rigorous examination of training the weights to various neural networks using backpropagation and differential evolution [23] was performed. Backpropagation is shown to be insufficient to train the neural networks. 12,000 runs of parallel differential evolution with various search parameters were executed on a high performance computing cluster, testing 15 different neural network designs. The results show that it is possible to have significant improvement over a baseline random noise estimator for airspeed and altitude. The best neural networks were cross-validated on flights they were not trained on, showing predictive ability and the potential to detect anomalous flights. These results provide a first step towards accurate prediction of FDR parameters, which could not only warn pilots of problematic flight behavior, but also be used to predict impending failures of engines and other hardware. This has the potential to reduce costs for maintaining general aviation fleets, and more importantly save lives.

2 Time-Series Prediction

Neural networks have been widely used for time series data prediction [7,29], however to the authors' knowledge, this is the first attempt to utilize them in predicting general aviation flight parameters. Our approach is similar to Khashei *et al.* [16] and Omer *et al.* [22], who utilize *residuals*, or *lags*, from linear data as additional inputs to the neural network. A significant difference is that this work uses multiple streams of input time-series data (airspeed, altitude, pitch and roll) to predict future values of each of those parameters; instead of prediction on single parameter time series data. This allows us to exploit dependencies between the input parameters.

2.1 Neural Network Design

Feed forward, Jordan and Elman networks were examined, each with 0, 1 and 2 layers of lag variables (as used in ARIMA time series models [28]); and 0 and 1 hidden layers, except for the Elman networks which require at least 1 hidden layer. The neural networks were used to predict one of the four input parameters, with examples shown in Figure 1. All the networks were fully connected between layers. The lag layers were added as additional input nodes to the neural network (one lag layer would add four additional input nodes, and two lag layers would add eight). The first order lag variables (Δ) are the difference between the current and previous timestep, *e.g.*:

$$\Delta_t(\text{Airspeed}) = \text{Airspeed}_t - \text{Airspeed}_{t-1} \quad (1)$$

where t is the current timestep and $t - 1$ is the previous timestep. The second order lag variables (Δ^2) are the difference between the current and previous first order lag variables, *e.g.*:

$$\Delta_t^2(\text{Airspeed}) = \Delta_t(\text{Airspeed}) - \Delta_{t-1}(\text{Airspeed}) \quad (2)$$

The following naming scheme was used to describe the various neural networks: *network_type/l(lag_Layers)/h(hidden_Layers)*. In the following scheme, *ff/1/1* would describe a feed forward network, with one lag layer and one hidden layer; while *jordan/1/2/h0* would describe a Jordan network with two lag layers (the first order lag variables and the second order lag variables) and no hidden nodes.

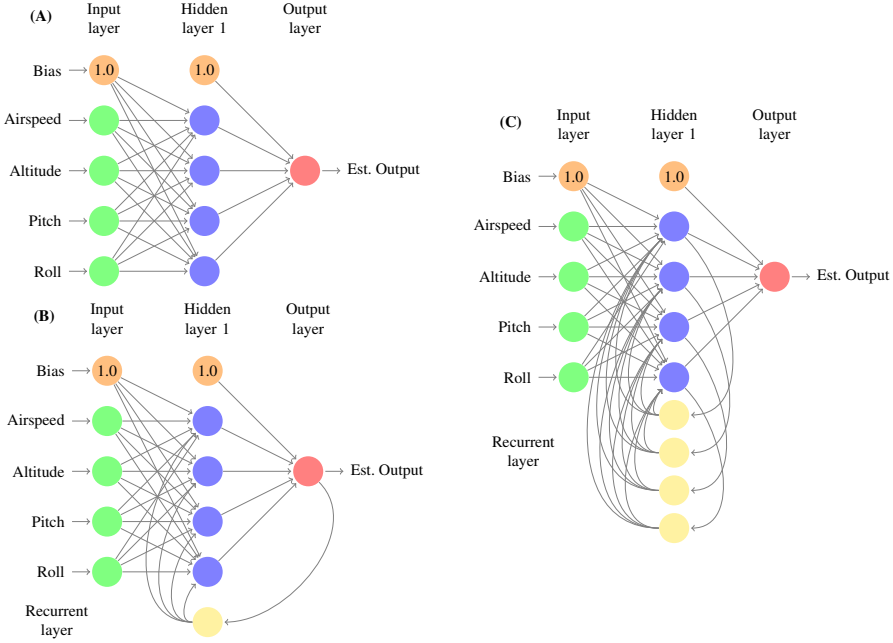


Fig. 1. Feed Forward (A), Jordan (B) and Elman (C) networks with a hidden layer and single output node. These networks were trained separately for each of the four possible outputs: airspeed, altitude, pitch and roll.

2.2 Objective Function

The neural networks were designed to predict the next second of flight data given the previous second of flight data and the first and second order lags variables, if used. To determine the optimal weights for the neural networks, the following objective function was used for both backpropagation and differential evolution:

$$f(w) = \frac{\sum_{t=0}^{n-1} |nn(I_t, \Delta_t, \Delta_t^2, w)_{output} - I_{t+1,output}|}{n - 1} \tag{3}$$

Where $f(w)$ is the objective function evaluated over the weights w . With I_t being the input at timestep t , the absolute difference between the output predicted by the neural network, $nn(\dots)_{output}$, and that value at the next time step, $I_{t+1,output}$, is calculated over every per second data reading ($0..n - 1$), given the input parameters, and input lags if used ($I_t, \Delta_t, \Delta_t^2$). This was then divided by the number of comparisons, $n - 1$. This produces the mean absolute error (MAE) for the target output for the entire flight.

2.3 Neural Network Bounds and Data Cleansing

The flight data required some cleaning for use, as it is stored as raw data from the flight data recorders uploaded to the NGAFID server and entered in the database as per second data. When a FDR turns on, some of the sensors are still calibrating or not immediately online, so the first minute of flight data can have missing and erroneous values. These initial recordings were removed from the data the neural networks were trained on. Further, the parameters had wide ranges and different units, *e.g.*, pitch and roll were in degrees, altitude was in meters and airspeed was in knots. These were all normalized to values between 0 and 1, where 0 was the minimum value recorded and 1 was the maximum value recorded for each parameter to remove bias.

Additionally, the recurrent neural networks needed to be bounded. The flights consisted of between 5412 and 5941 per second recordings (over an hour and a half of per second data). This led to a problem where poor weights to the recurrent layer resulted in the fitness growing beyond the bounds of double precision. To alleviate this problem, the values for the recurrent nodes were bounded between -2 and 3 which eliminated overflow errors. Lastly, the weights for the neural networks were all bounded between -1.5 and 1.5 to limit the search space of the evolutionary algorithms and initial values for backpropagation.

3 Parallel Evolutionary Algorithms

A wide range of evolutionary algorithms have been examined for different distributed computing environments. Generally, these fall into three categories: single population (panmictic, coarse-grained); multi-population (island, medium-grained); or cellular (fine-grained); as classified by Cantu-Paz [6]. These various approaches have different effects on the explorative and exploitative properties of the evolutionary algorithms [26], with smaller subpopulations allowing faster exploitation of their search subspaces.

Given the scale of the data in the NGAFID, and the potential complexity of examining complex neural networks over per-second flight data, a package requiring easy use of high performance computing resources was required. While there exist some standardized evolutionary algorithms packages [2,5,27,17], as well as those found in the R programming language [20,3] and MATLAB [18], they do not easily lend themselves towards use in high performance computing environments.

This work utilizes the Toolkit for Asynchronous Optimization (TAO), which is used by the MilkyWay@Home volunteer computing to perform massively distributed evolutionary algorithms on tens of thousands of volunteered hosts [10,11,8]. It is implemented in C and MPI, allowing easy use on clusters and supercomputers, and also provides support for systems with multiple graphical processing units. Further, TAO has shown that performing evolutionary algorithms asynchronously can provide significant improvements to performance and scalability over iterative approaches [25,9]. Its code is also open source and freely available on GitHub, allowing easy use and extensibility¹.

¹ <https://github.com/travisesell/tao>

4 Results

4.1 Runtime Environment

All results were gathered using a Beowulf HPC cluster with 32 dual quad-core compute nodes (for a total of 256 processing cores). Each compute node has 64GBs of 1600MHz RAM, two mirrored RAID 146GB 15K RPM SAS drives, two quad-core E5-2643 Intel processors which operate at 3.3Ghz, and run the Red Hat Enterprise Linux (RHEL) 6.2 operating system. All 32 nodes within the cluster are linked by a private 56 gigabit (Gb) InfiniBand (IB) FDR 1-to-1 network. The code was compiled and run using MVAPICH2-x [13], to allow highly optimized use of this network infrastructure.

Each run of a differential evolution/neural network combination was submitted as a single job, allocating 32 processes across 4 nodes, with 1 master process to handle generating new individuals and updating the population, and 31 to handle the objective function calculations. The asynchronous differential evolution implementation from TAO was used to perform this optimization.

The results for optimizing all combinations of the neural networks and input parameters to differential evolution required 12000 jobs, approximately 4,800 hours of single CPU compute time. These jobs had a minimum runtime of 2.3 seconds, a maximum runtime of 278 seconds and an average runtime of 45.1 seconds. As such, utilizing parallel differential evolution on a HPC system enabled running these jobs in a reasonable amount of time, taking approximately a week given shared resources with other users of the cluster.

4.2 Evaluation Metrics

A random noise estimator (RNE), which uses the previous value as the prediction for the next value, $prediction(t_{i+1}) = t_i$, was chosen as a baseline comparison, as it represents the best predictive power that can be achieved for random time series data. If the neural networks did not improve on this, then the results would have been meaningless and potentially indicate that the data is too noisy (given weather and other conditions) for prediction. Additionally, it provides a good baseline in that it is easy for neural networks to represent the RNE: all weights can be set to 0, except for a single path from the path from the corresponding input node to the output node having weights of 1. Because of this, it also provides a good test of the correctness of the global optimization techniques, at the very least they should be able to train a network as effective as a RNE; however local optimization techniques (such as backpropagation) may not reach this if the search area is non-convex and the initial starting point does not lead to a good minimum.

4.3 Infeasibility of Backpropagation

Backpropagation was evaluated using both stochastic gradient descent (GD) and conjugate gradient descent (CGD) on flight 13588. Stochastic GD and CGD were run 20 different times for the networks described in Section 2. Figure 2 presents the range of fitnesses found for each backpropagation and network combination. In all cases, stochastic GD and CGD performed worse than RNE, demonstrating the challenging and non-convex nature of this search area. Accuracy further decreased and became

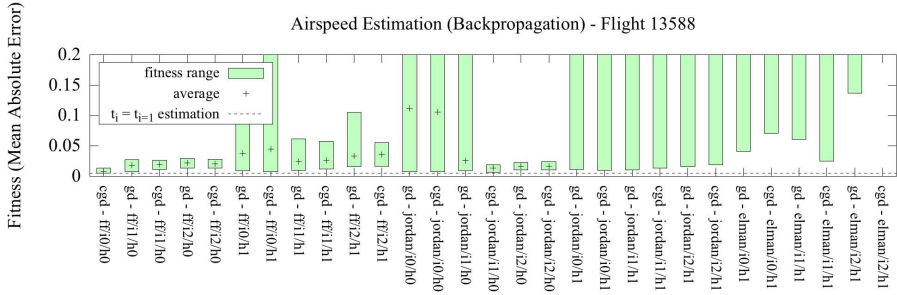


Fig. 2. Histograms for the estimation of airspeed for flight 13588 using backpropagation via gradient descent (GD) and conjugate gradient descent (CGD). The dashed line shows the baseline estimation for random noise, using the value at timestep t_i as the prediction for the value at timestep t_{i+1} .

more variable as neural networks became more complicated. Stochastic GD and CGD performed similarly for altitude, pitch, and roll.

Backpropagation was also evaluated using GD and CGD starting from a neural network which simulates a RNE, to address the case of all the initial starting positions of stochastic GD and CGD being poor. However, even the neural networks representing RNE for each network type were close to a local minima. Using backpropagation starting from a RNE, Airspeed was only improved upon at best by 0.018%, altitude improved on by 0.5%, pitch improved on by 0.025%, and roll improved upon by 0.022%.

4.4 Neural Network Optimization

Given previous experience training these neural networks, and in order to perform the analysis of more types of neural networks, we limited the differential evolution options to *de/best/3/bin* and *de/rand/3/bin* (differential evolution with best or random parent selection, 3 pairs, and binary recombination – for a detailed description of differential evolution variants see [19]), with a population size of 500; as these settings had been shown to provide good convergence rates and effective solutions in the past.

Five flights (13588, 15438, 17269, 175755 and 24335) were used for analysis, and neural networks were trained to predict each of the four flight data parameters (airspeed, altitude, pitch and roll) used as input. Each differential evolution strategy and neural network combination was run 20 times, for a total of 12000 runs (2 DE strategies x 15 neural networks x 4 output parameters x 5 flights x 20 runs each). The parallel differential evolution was terminated after 15,000,000 total objective functions had been evaluated, or the best fitness in the population had not improved after 1,000 generations.

Table 1 show rankings for the different neural networks in terms of how well they predicted the given output. The ranks are what order the network came in, in terms of prediction; *i.e.*, a 1 means the neural network gave the best fitness for one of the flights, a 2 means it gave the second best fitness, and so on. Ties (in some cases multiple networks had the same best fitness) were given the same rank, so in some cases there are more than 5 of each rank. The rank column provides the average rank the network came in across all flights, and the avg. evals column gives the average number of evaluations it took for the differential evolution to terminate.

Table 1. Neural Network Rankings

Airspeed				Pitch			
Network	Rank	Ranks	Avg. Evals	Network	Rank	Ranks	Avg. Evals
elman/i0/h1	19.9	10, 16, 17, 17, 19, 20, 23, 25, 26, 26	2667750	elman/i0/h1	13.0	6, 10, 10, 12, 12, 12, 12, 16, 20, 20	3543250
elman/i1/h1	14.0	4, 10, 11, 12, 13, 16, 18, 18, 19, 19	3334000	elman/i1/h1	11.6	1, 10, 10, 11, 11, 11, 13, 14, 16, 19	4104750
elman/i2/h1	6.7	1, 3, 3, 4, 5, 6, 7, 9, 14, 15	4225500	elman/i2/h1	6.2	1, 1, 2, 2, 3, 7, 8, 9, 11, 18	4793500
ff/i0/h0	22.6	21, 21, 22, 22, 23, 23, 23, 24, 24, 24	8300000	ff/i0/h0	23.8	23, 23, 23, 23, 24, 24, 24, 24, 25, 25	905000
ff/i0/h1	24.1	22, 23, 23, 24, 24, 24, 25, 25, 25, 26	10500000	ff/i0/h1	25.3	24, 24, 25, 25, 25, 25, 26, 26, 26, 27	1105000
ff/i1/h0	13.0	11, 11, 12, 12, 13, 13, 14, 14, 15, 15	15150000	ff/i1/h0	16.4	15, 15, 15, 15, 17, 17, 17, 17, 18, 18	1200000
ff/i1/h1	14.5	12, 13, 13, 14, 14, 15, 15, 16, 16, 17	1720000	ff/i1/h1	17.9	16, 16, 17, 17, 18, 18, 19, 19, 20	1675000
ff/i2/h0	6.6	5, 5, 5, 6, 6, 6, 8, 8, 8, 9	2615000	ff/i2/h0	5.2	3, 3, 5, 5, 6, 6, 6, 6, 6, 6	2595000
ff/i2/h1	8.3	6, 7, 7, 7, 8, 8, 9, 10, 10, 11	2225000	ff/i2/h1	5.4	1, 2, 4, 5, 5, 7, 7, 7, 8, 8	2045000
jordan/i0/h0	19.8	19, 19, 19, 19, 20, 20, 20, 20, 21, 21	1573750	jordan/i0/h0	20.7	20, 20, 20, 20, 21, 21, 21, 21, 22	1372500
jordan/i0/h1	21.9	20, 20, 21, 21, 22, 22, 22, 24, 26	2551750	jordan/i0/h1	22.3	21, 21, 22, 22, 22, 23, 23, 23, 24	2449750
jordan/i1/h0	5.0	2, 2, 3, 3, 3, 3, 5, 6, 11, 12	6897750	jordan/i1/h0	12.5	9, 10, 12, 12, 13, 13, 14, 14, 15	4874750
jordan/i1/h1	15.5	9, 13, 14, 15, 16, 17, 17, 18, 18, 18	3712000	jordan/i1/h1	15.4	13, 13, 14, 14, 15, 15, 16, 17, 18, 19	3640000
jordan/i2/h0	1.6	1, 1, 1, 1, 1, 2, 2, 2, 2, 3	11695500	jordan/i2/h0	3.1	1, 2, 2, 3, 3, 3, 4, 4, 4, 5	9692500
jordan/i2/h1	7.4	4, 4, 4, 5, 7, 8, 9, 10, 11, 12	5078750	jordan/i2/h1	7.6	4, 5, 6, 7, 8, 8, 9, 9, 11	5269750

Altitude				Roll			
Network	Rank	Ranks	Avg. Evals	Network	Rank	Ranks	Avg. Evals
elman/i0/h1	22.4	20, 21, 21, 22, 22, 22, 23, 23, 24, 26	2629000	elman/i0/h1	18.3	4, 16, 17, 17, 18, 19, 21, 23, 24, 24	3096000
elman/i1/h1	16.6	11, 13, 13, 13, 16, 17, 17, 17, 24, 25	3704000	elman/i1/h1	13.7	1, 2, 14, 15, 16, 16, 17, 18, 18, 20	3882500
elman/i2/h1	16.2	12, 12, 13, 13, 15, 16, 18, 19, 19, 25	4583750	elman/i2/h1	5.0	1, 1, 1, 1, 2, 2, 3, 9, 14, 16	4707750
ff/i0/h0	20.4	18, 18, 19, 19, 20, 20, 21, 21, 24, 24	7800000	ff/i0/h0	23.8	22, 22, 23, 23, 24, 24, 25, 25, 25, 25	710000
ff/i0/h1	21.7	19, 19, 20, 20, 21, 22, 22, 23, 25, 26	12050000	ff/i0/h1	25.3	23, 24, 24, 25, 25, 26, 26, 26, 27, 27	1090000
ff/i1/h0	8.6	7, 7, 8, 8, 8, 8, 10, 10, 10, 10	1635000	ff/i1/h0	11.6	8, 8, 11, 11, 11, 11, 13, 13, 15, 15	1435000
ff/i1/h1	10.0	8, 9, 9, 9, 10, 11, 11, 12, 12	1705000	ff/i1/h1	13.1	9, 10, 12, 12, 13, 13, 14, 15, 16, 17	1655000
ff/i2/h0	3.2	2, 2, 3, 3, 3, 3, 4, 4, 4, 4	2060000	ff/i2/h0	5.4	4, 4, 4, 4, 5, 6, 6, 6, 7, 8	2355000
ff/i2/h1	4.2	3, 3, 4, 4, 4, 4, 4, 5, 5, 6	2335000	ff/i2/h1	7.2	5, 5, 6, 6, 7, 7, 8, 9, 9, 10	2375000
jordan/i0/h0	17.1	15, 16, 16, 16, 16, 17, 18, 20, 21	1153500	jordan/i0/h0	19.7	18, 18, 19, 19, 20, 20, 21, 21, 21	1522500
jordan/i0/h1	15.9	14, 14, 14, 15, 15, 15, 17, 18, 18, 19	2425500	jordan/i0/h1	21.5	19, 20, 21, 21, 22, 22, 22, 23, 23	2387500
jordan/i1/h0	6.6	6, 6, 6, 6, 6, 6, 7, 7, 8, 8	4860500	jordan/i1/h0	10.0	7, 7, 9, 10, 10, 10, 10, 13, 14	3954750
jordan/i1/h1	10.7	7, 9, 9, 10, 10, 11, 11, 12, 14, 14	4421750	jordan/i1/h1	14.5	11, 11, 12, 13, 14, 15, 15, 17, 18, 19	3740750
jordan/i2/h0	1.3	1, 1, 1, 1, 1, 1, 1, 2, 2, 2	11706250	jordan/i2/h0	3.3	2, 2, 2, 3, 3, 3, 3, 4, 5, 6	6838000
jordan/i2/h1	5.1	2, 3, 3, 5, 5, 5, 6, 7, 7, 8	6343750	jordan/i2/h1	7.7	3, 4, 5, 7, 7, 8, 8, 11, 12, 12	5292750

These results show that there is no clear cut best neural network for prediction of all these flight parameters. However, the Jordan and Elman networks tend to outperform the feed forward neural networks; and in most cases adding input lags improves the predictive ability of the network. Except in the case of the Elman networks, adding a hidden layer does not seem to provide much benefit (perhaps due to the increased difficulty of optimization). Generally speaking, it seems that Jordan networks (with 2 input lags and no hidden layer) perform the best for predicting altitude and airspeed, while Elman networks (with 2 input lags and 1 hidden layer) perform the best for pitch and roll. Also worth noting is that when the Jordan networks perform the best, they take significantly longer to converge to a solution. Given this, it may be the case that the Elman networks are either converging prematurely or getting stuck. This question does beg further examination, as the more complicated neural networks should theoretically be able to provide more predictive power.

4.5 Cross Validation

In order to gauge the predictive power of the trained neural networks, the best found neural networks for each flight and output parameter were cross validated against the other flights. In Table 2, each row shows the best found neural network for each flight. The first row shows the random noise estimation (RNE), for baseline comparison. Each column in that row shows the mean absolute error (MAE) for the neural network trained for the flight specified flight against all the other flights. The bold values show the MAE

where the input flight was the flight compared against; while italicized values show where the neural network performed worse than the RNE.

These results show that the trained neural networks have predictive parameters of other flights. They also show a dramatic difference in predictive ability for the different output parameters. Excluding the neural networks trained on flight 17269, predicted airspeed showed a 10-12% improvement over RNE, altitude showed near 70% improvement, while pitch and roll were much lower at 5-7% and 0.5-3%, respectively. Most of the trained neural networks were able to improve over RNE for all the other flights that they were not trained on. Further, the predictions are fairly accurate. As the input and output parameters were normalized between 0 and 1, the mean average error is also the average percentage error for the prediction. Airspeed predictions were around 0.6% error, altitude predictions were around 0.08% error, pitch was around 1.5% error, and roll was around 2% error.

These results lead to some interesting findings: first, the four parameters used (altitude, airspeed, pitch and roll) are probably not sufficient for prediction of pitch and roll, however they do provide good inputs for predicting airspeed and especially altitude. Using additional input parameters should allow better prediction for these values. Second, using this cross validation it appears that flight 17269 is an outlier, especially in pitch, as it was 50% worse than RNE in predicting pitch from the other flights. These findings present the possibility that it may be possible to determine atypical flights utilizing trained neural networks and potentially identify problematic parameters.

5 Conclusions and Future Work

This work provides an extensive analysis of flight parameter estimation using various neural networks and input parameters to differential evolution. The neural networks were trained on data recorded during actual student flights, and consist of noisy, realistic general aviation flight data. Results show that while backpropagation is unable to provide much improvement over a random noise estimator (RNE), parallel differential evolution can provide strong predictions of airspeed (10% better than RNE) and altitude (70% better than RNE). These results were also fairly accurate, ranging between 0.08% accuracy for altitude to 2% accuracy for roll. Cross validation indicate that the trained neural networks have predictive ability, as well as the potential to act as overall descriptors of the flights. The trained neural networks could be used to detect anomalous flights, and even determine which flight parameters are causing the anomaly (*e.g.*, pitch in flight 17269).

For future work, how well the neural networks can be trained using particle swarm optimization [15] is of particular interest; as well as using autoencoders and other deep learning strategies [4], or hybrid strategies with genetic algorithms or ant colony optimization to evolve the structure of more complicated neural networks. Performing a grid search over potential evolutionary algorithm parameters is also suboptimal, which can be improved on by using hyperparameter optimization [14] or other metaheuristics. Further, training the neural networks over groups of flights could potentially improve their overall predictive ability as well as minimize overtraining.

The National General Aviation Flight Database (NGAFID) provides an excellent data source for researching evolutionary algorithms, machine learning and data mining.

Table 2. Cross Validation for All Flight Parameters and Flights

Airspeed						
Method	13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$	0.00512158	0.00316859	0.00675531	0.00508229	0.00575537	0.0
13588 elman/i2/h1	0.00472131	0.00250284	0.00656991	0.00465581	0.00495454	10.78%
15438 jordan/i2/h0	0.00500836	0.00218919	0.0067222	0.00480868	0.00498588	10.47%
17269 jordan/i2/h0	<i>0.00513133</i>	0.0027844	0.00620534	0.00505878	0.00552826	4.90%
175755 jordan/i2/h0	0.0047884	0.00240848	0.00643301	0.00459774	0.00498664	11.63%
24335 jordan/i2/h0	0.00487011	0.00226412	0.00666179	0.00471104	0.00485888	11.54%

Altitude						
Method	13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$	0.00138854	0.00107117	0.00200011	0.00137109	0.00192345	0.0
13588 jordan/i2/h0	0.000367535	0.000305193	0.000895711	0.000399587	0.000485329	69.18%
15438 jordan/i2/h0	0.000394097	0.000263834	0.000837357	0.0004203	0.00048358	69.87%
17269 jordan/i2/h0	0.000702832	0.000765161	0.000801323	0.000694245	0.000846411	48.65%
175755 jordan/i2/h0	0.00037486	0.0003003	0.000883877	0.000390743	0.00048446	69.42%
24335 jordan/i2/h0	0.000380966	0.000281196	0.000906039	0.000404582	0.000468267	69.43%

Pitch						
Method	13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$	0.0153181	0.010955	0.0148046	0.0161251	0.0173269	0.0
13588 elman/i1/h1	0.014918	0.0100763	0.0147712	0.01514	0.0160249	4.90%
15438 jordan/i2/h0	<i>0.0163609</i>	0.00881572	<i>0.0159061</i>	0.0150275	0.015552	4.47%
17269 elman/i2/h1	<i>0.0199653</i>	<i>0.0249148</i>	0.0142671	<i>0.0199625</i>	<i>0.0291537</i>	-49.24%
175755 ff/i2/h1	<i>0.0153644</i>	0.00917981	<i>0.0148751</i>	0.0145228	0.0153566	7.35%
24335 elman/i2/h1	<i>0.0157302</i>	0.00911826	<i>0.0160291</i>	0.014868	0.0149484	5.47%

Roll						
Method	13588	15438	17269	175755	24335	Improvement
$t_{i+1} = t_i$	0.0158853	0.00604479	0.0204441	0.012877	0.0192648	0.0
13588 elman/i2/h1	0.0154541	0.00587058	<i>0.0206536</i>	0.0127999	0.0182611	2.08%
15438 elman/i2/h1	<i>0.0164341</i>	0.00544584	<i>0.0217141</i>	<i>0.0129252</i>	0.0176981	1.60%
17269 elman/i1/h1	0.0157483	<i>0.00613587</i>	0.0201234	<i>0.0129124</i>	0.0184769	0.95%
175755 elman/i2/h1	0.0156503	0.00573676	<i>0.0205158</i>	0.0125207	0.017983	3.13%
24335 elman/i2/h1	<i>0.0163245</i>	0.00578885	<i>0.0215668</i>	<i>0.0131439</i>	0.0174324	0.68%

Further analysis of these flights along with more advanced prediction methods will enable more advanced flight sensors, which could prevent accidents and save lives; which is especially important in the field of general aviation as it is has the highest accident rates within civil aviation [21]. As many of these flights also contain per-second data of various engine parameters, using similar predictive methods it may become possible to detect engine and other hardware failures, aiding in the maintenance process. This work presents an initial step towards making general aviation safer through machine learning and evolutionary algorithms.

References

1. Aircraft Owners and Pilots Association (AOPA) (January 2014)
2. Arenas, M., Collet, P., Eiben, A.E., Jelasity, M., Merelo, J.J., Paechter, B., Preuß, M., Schoenauer, M.: A framework for distributed evolutionary algorithms. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 665–675. Springer, Heidelberg (2002)
3. Bartz-Beielstein, T.: SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. arXiv preprint arXiv:1006.4645 (2010)

4. Bengio, Y.: Learning deep architectures for ai. *Foundations and trends® in Machine Learning* 2(1), 1–127 (2009)
5. Cahon, S., Melab, N., Talbi, E.-G.: Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics* 10(3), 357–380 (2004)
6. Cantu-Paz, E.: A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis* 10(2), 141–171 (1998)
7. Crone, S.F., Hibon, M., Nikolopoulos, K.: Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting* 27(3), 635–660 (2011)
8. Desell, T.: *Asynchronous Global Optimization for Massive Scale Computing*. PhD thesis, Rensselaer Polytechnic Institute (2009)
9. Desell, T., Anderson, D., Magdon-Ismael, M., Heidi Newberg, B.S., Varela, C.: An analysis of massively distributed evolutionary algorithms. In: *The 2010 IEEE Congress on Evolutionary Computation (IEEE CEC 2010)*, Barcelona, Spain (July 2010)
10. Desell, T., Szymanski, B., Varela, C.: Asynchronous genetic search for scientific modeling on large-scale heterogeneous environments. In: *17th International Heterogeneity in Computing Workshop*, Miami, Florida (April 2008)
11. Desell, T., Varela, C., Szymanski, B.: An asynchronous hybrid genetic-simplex search for modeling the Milky Way galaxy using volunteer computing. In: *Genetic and Evolutionary Computation Conference (GECCO)*, Atlanta, Georgia (July 2008)
12. Elias, B.: *Securing general aviation*. DIANE Publishing (2009)
13. Huang, W., Santhanaraman, G., Jin, H.-W., Gao, Q., Panda, D.K.: Design of high performance MVAPICH2: MPI2 over InfiniBand. In: *Sixth IEEE International Symposium on Cluster Computing and the Grid, CCGRID 2006*, vol. 1, pp. 43–48. IEEE (2006)
14. Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient approach for assessing hyperparameter importance. In: *Proc. of ICML 2014* (to appear, 2014)
15. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: *IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948 (1995)
16. Khashei, M., Bijari, M.: A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing* 11(2), 2664–2675 (2011)
17. Lukaszewicz, M., Glaß, M., Reimann, F., Teich, J.: Opt4j: a modular framework for metaheuristic optimization. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011*, pp. 1723–1730. ACM, New York (2011)
18. MathWorks. *Global optimization toolbox* (March 2013) (accessed online)
19. Mezura-Montes, E., Velazquez-Reyes, J., Coello Coello, C.C.A.: Modified differential evolution for constrained optimization. In: *IEEE Congress on Evolutionary Computation 2006, CEC 2006*, Vancouver, BC, pp. 25–32 (July 2006)
20. Mullen, K., Ardia, D., Gil, D., Windover, D., Cline, J.: Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software* 40(6), 1–26 (2011)
21. National Transportation Safety Board (NTSB) (2012)
22. Ömer Faruk, D.: A hybrid neural network and arima model for water quality time series prediction. *Engineering Applications of Artificial Intelligence* 23(4), 586–594 (2010)
23. Schwefel, H.-P.: *Evolution and Optimization Seeking*. John Wiley & Sons, New York (1995)
24. Shetty, K.I.: *Current and historical trends in general aviation in the United States*. PhD thesis, Massachusetts Institute of Technology Cambridge, MA 02139 USA (2012)
25. Szymanski, B.K., Desell, T., Varela, C.: The effects of heterogeneity on asynchronous panmictic genetic search. In: *Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS*, vol. 4967, pp. 457–468. Springer, Heidelberg (2008)
26. Črepinšek, M., Liu, S.-H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.* 45(3), 35:1–35:33 (2013)

27. Ventura, S., Romero, C., Zafra, A., Delgado, J.A., Hervás, C.: Jelec: a java framework for evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications* 12(4), 381–392 (2008)
28. Wei, W.W.-S.: *Time series analysis*. Addison-Wesley, Redwood City (1994)
29. Zhang, G.P.: *Neural networks for time-series forecasting*. In: *Handbook of Natural Computing*, pp. 461–477. Springer (2012)

Random Partial Neighborhood Search for University Course Timetabling Problem

Yuichi Nagata¹ and Isao Ono²

¹ Education Academy of Computational Life Sciences,
Tokyo Institute of Technology, Japan

² Institute of Technology and Science,
The University of Tokushima, Japan

nagata@is.tokushima-u.ac.jp, isao@dis.titech.ac.jp

Abstract. We propose an tabu search algorithm using an candidate list strategy with random sampling for the university course timetabling problem, where the neighborhood size can be adjusted by a parameter *ratio*. With this framework, we can control the trade-off between exploration and exploitation by adjusting the neighborhood size. Experimental results show that the proposed algorithm outperforms state-of-the-art algorithms when the neighborhood size is set properly.

1 Introduction

To solve optimization problems that are computationally intractable, heuristic (approximation) algorithms have been widely used for finding nearly optimal solutions in a reasonable computation time. In particular, neighborhood search is a wide class of heuristic algorithms, where the current solution is iteratively moved to a solution in the neighborhood at each iteration.

Crucial issues in the design of an effective neighborhood search algorithm are the choices of the neighborhood structure and search strategy. The neighborhood is defined as a set of solutions that are obtained typically by performing prearranged local modifications on the current solution. The choice of the neighborhood structure is very important because it directly affects the fitness landscape. The choice of the search strategy is also important because it controls the trade-off between exploration and exploitation of the search. A lot of search strategies have been proposed in the literature, ranging from simple hill-climbing, simulated annealing, tabu search, guided local search [7] to more sophisticated ones, where the trade-off between exploration and exploitation can be controlled by their specific parameters (e.g. *temperature* for SA, *tabu tenure* for TS, and λ value for GLS).

Most of the neighborhood search algorithms use a fixed neighborhood structure (sometimes a composite of several neighborhoods) throughout the search while controlling the trade-off between exploration and exploitation by the search strategy. Contrary to these algorithms, candidate list strategies [3] consider only a subset of a predefined full neighborhood at each iteration in order to reduce

the computational effort. The simplest way of introducing candidate list strategy is to define a neighborhood as a randomly selected part of a predefined full neighborhood. Apart from the effect of reducing the computational effort, this candidate list strategy is useful for diversifying the search. With this strategy, we can control the trade-off between exploration and exploitation by adjusting the ratio of the size of the partial neighborhood to that of the full neighborhood; the smaller the neighborhood size is, the more diverse is the search. This is because the quality of a solution accepted in the partial neighborhood becomes worse with decreasing the neighborhood size (if a solution must be accepted at each iteration).

In this paper, we incorporate the candidate list strategy with random sampling into a tabu search framework to construct an effective neighborhood search algorithm for the university course timetabling problem (UCTP), and in this paper we call this algorithm random partial neighborhood search (RPNS). In addition, we analyze the effect of changing the neighborhood size on the performance. Experimental results on the well-studied benchmark set of Socha *et al.* [6] show that the RPNS algorithm outperforms the state-of-the-art algorithms [1][4][5][2] when the neighborhood size is set properly.

The remainder of this paper is organized as follows. The definition of the UCTP is described in Section 2. The framework of RPNS is presented in Section 3. The computational analysis of the RPNS algorithm and the performance comparison with state-of-the-art algorithms are presented in Section 4. Conclusions are presented in Section 5.

2 The University Course Timetabling Problem

Several formulations of the UCTP have been proposed in the literature, and we select the one proposed by Socha *et al.* [6] because the corresponding benchmark set, which is available at <http://iridia.ulb.ac.be/~msampels/tt.data/>, have been intensively tackled by many algorithms.

Let $E = \{e_1, e_2, \dots, e_N\}$ denote a set of N events, $T = \{t_1, t_2, \dots, t_K\}$ a set of K timeslots ($K = 45$, 5 days of 9 hours), and $R = \{r_1, r_2, \dots, r_L\}$ a set of L rooms, $S = \{s_1, s_2, \dots, s_M\}$ a set of M students. For each event, the students who attend this event and a set of the rooms that meet requirements for this event¹ are known.

A timetable is represented as an assignment of the events to the timeslots and rooms. A timetable is said to be *feasible* if it satisfies the following hard constraints (H1-H3).

H1: No student attends more than one event in the same timeslot.

H2: Each event must be assigned to a room that meets the requirements for this event.

H3: Only one event can be assigned to each room at any timeslot.

¹ In the original benchmark data set, this information is not explicitly given, but it is easily obtained from the given data.

The objective is to find a feasible timetable that minimizes the total violations of the soft constraints (S1-S3) described below.

S1: A student has an event scheduled in the last timeslot of a day.

S2: A student has more than two consecutive events ².

S3: A student has only one event on a day.

More formally, the total violations of the soft constraints for a timetable x is defined as follows:

$$f(x) = f_1(x) + f_2(x) + f_3(x),$$

where $f_i(x)$ ($i = 1, 2, 3$) is the number of the occurrence of constraint violations in terms of soft constraint S_i .

3 Solution Method

As used in some of the previous works, we employ a two-stage approach where a feasible timetable is constructed in the first stage (not main focus of this paper) and then the total violation of the soft constraints is minimized in the second stage (main focus of this paper) while maintaining the feasibility. We first present the RPNS algorithm that is used in the second stage and then describe the outline of the first stage.

3.1 Random Partial Neighborhood Search

Neighborhoods. We first define the two neighborhoods \mathcal{N}_1 and \mathcal{N}_2 , which are widely used in neighborhood searches for various timetabling problems.

$\mathcal{N}_1(x)$: A set of the feasible timetables that are obtained from a timetable x by moving an event e to another timeslot-room pair (t, r) for all possible (e, r, t) combinations.

$\mathcal{N}_2(x)$: A set of the feasible timetables that are obtained from a timetable x by exchanging the timeslots of two events (e_1, e_2) for all possible (e_1, e_2) combinations, where a change of the room is allowed unless the room is not occupied by other events.

The neighborhood \mathcal{N}_2 is slightly different from the standard swap neighborhood because the standard swap move is to exchange the assignment of timeslot-room pairs between two events (or exchange is allowed only if two events are assigned to the same room). This modification is reasonable because the \mathcal{N}_2 neighborhood is more flexible in the assignment of rooms and it scarcely increases the computational effort to evaluate all possible moves in the neighborhood.

To vary the neighborhood size, we introduce two neighborhoods $\mathcal{N}_1(x, ratio)$ and $\mathcal{N}_2(x, ratio)$ as partial neighborhoods of $\mathcal{N}_1(x)$ and $\mathcal{N}_2(x)$, respectively, where *ratio* ($0 \leq ratio \leq 1$) is a parameter that specifies the ratio of the size of the partial neighborhood to that of the full neighborhood. These neighborhoods are defined as follows.

² Two events in different days are not regarded as consecutive. If the number of consecutive events is s (≥ 3), the number of violations caused by these events is $s - 2$.

$\mathcal{N}_1(x, ratio)$: A set of the feasible timetables that are obtained from a timetable x by moving an event e to another timeslot-room pair (t, r) for $e \in I_1$ and all possible (r, t) combinations, where I_1 is defined by randomly selecting $\lceil ratio \times N \rceil$ elements of E .

$\mathcal{N}_2(x, ratio)$: A set of the feasible timetables that are obtained from a timetable x by exchanging the timeslots of two events (e_1, e_2) for $e_1 \in I_2$ and $e_2 \in E (e_1 < e_2)$ (i.e., at least one of the two events is selected from I_2) where I_2 is defined by randomly selecting $\lceil \frac{1}{2} \left\{ 2N - 1 - \sqrt{4N(N - 1)(1 - ratio) + 1} \right\} \rceil$ ($= N_2$) element of E . The change of room is allowed for both events e_1 and e_2 unless the room is not occupied by other events.

Note that N_2 in the definition of $\mathcal{N}_2(x, ratio)$ neighborhood is determined such that the size of $\mathcal{N}_2(x, ratio)$ neighborhood divided by the size of $\mathcal{N}_2(x)$ neighborhood is equal to $ratio$. In fact, N_2 is obtained by solving the following equation: $N_2N - \frac{N_2(N_2+1)}{2} = ratio \times \frac{N(N-1)}{2}$.

Algorithm. The idea of using the random partial neighborhood is incorporated into tabu search (TS) to construct a RPNS algorithm, which is presented in Algorithm 1. Before starting iterations, the current timetable x and the current best timetable x_{best} are initialized with an input feasible timetable (line 1). At each iteration, the best non-tabu solution x' , which will become the next current timetable, is selected from the union of the two partial neighborhoods $\mathcal{N}_1(x, ratio)$ and $\mathcal{N}_2(x, ratio)$ where the selection of a tabu solution is forbidden (line 3). Tabu solutions are defined as follows. If an event e is moved using \mathcal{N}_1 neighborhood (or two events e_1 and e_2 are swapped using \mathcal{N}_2 neighborhood), event e (or two events e_1 and e_2) is regarded as “tabu event” during the subsequent T iterations, where T is a parameter called tabu tenure. At each iteration, a timetable obtained by moving an tabu event using \mathcal{N}_1 neighborhood or by swapping two tabu events using \mathcal{N}_2 neighborhood is regarded as a tabu solution. In addition, the aspiration criterion is considered where a solution that improves the current best solution x_{best} is regarded as a non-tabu solution. After selecting the best non-tabu solution x' , the current solution x and current best solution x_{best} (if necessary) are updated by x' (line4). Iterations are repeated until the total number of iterations reaches a given maximum number of iterations $iterMax$ (lines 2 and 5), and the current best timetable x_{best} is returned (line 7).

We should note that it is also possible to define tabu solutions based on the attribute of event-timeslot pairs (e.g. an event is forbidden from moving back to timeslot t during the subsequent T iterations after this event is moved from timeslot t in a previous iteration) rather than the attribute of only events. However, we decided to employ our definition because we confirmed that the performance of RPNS with the tabu definition based on only events was slightly better than that with tabu definition based on event-timeslot pairs and the former one is simpler.

Algorithm 1. TABU-SEARCH(x_{input})

```

1: Set  $x := x_{input}$ ,  $x_{best} := x_{input}$  and  $iter := 0$ ;
2: while ( $iter \leq iterMax$ ) do
3:   Select a best non-tabu solution  $x' \in \mathcal{N}_1(x, ratio) \cup \mathcal{N}_2(x, ratio)$  (the aspiration
   criterion is considered);
4:   Update  $x := x'$  and  $x_{best} := x'$  (if  $x'$  is better than  $x_{best}$ );
5:   Set  $iter := iter + 1$ ;
6: end while
7: return  $x_{best}$ ;

```

3.2 Construction of an Initial Feasible Timetable

To construct an initial feasible timetable in the first stage, we use an algorithm similar to tabu search (TS) where a current solution is represented as a partial timetable during the course of the search. Here, a partial timetable refers to an incomplete timetable in which one or more events are not scheduled. A partial timetable is regarded as feasible if the scheduled events satisfy all hard constraints. We evaluate the quality of partial timetables by the number of the unscheduled events (small number is better). The unscheduled events are stored in a list called ejection list (EL).

The procedure is started by initializing both the current timetable x and the current best timetable x_{best} with an empty timetable and EL with all events in a random order. At each iteration, an event e_{in} is popped from EL , and an attempt is made to schedule the selected event into the current (partial) timetable x without violating any hard constraints. Let $\mathcal{N}_{in_out}(e_{in}, x)$ be a set of the feasible (partial) timetables that are obtained from x by assigning e_{in} to a timeslot-room pair and ejecting the conflicting events caused by the insertion of e_{in} (no event is ejected if no hard constraint violation occurs) in all possible ways. Note that e_{in} must be assigned to a room that satisfies the requirement for this event (hard constraint H2), while the violation of the hard constraints H1 and H3 caused by the insertion of e_{in} can be resolved by ejecting the conflicting events in the same timeslot. The next current timetable x' is selected from $\mathcal{N}_{in_out}(e_{in}, x)$ so that the number of the ejecting events is minimized. In practice, the idea of tabu search is incorporated to diversity the search; the selection of a tabu solution is forbidden where a timetable obtained by assigning e_{in} to a timeslot t is regarded as a tabu solution during the subsequent $T(= 100)$ iterations after e_{in} is assigned to a timeslot t in the previous iteration. In addition, the aspiration criterion is considered, where a solution that improves the current best solution x_{best} is always regarded as a non-tabu solution. After the selection of the next current timetable, push the ejected events into EL if one or more events are ejected, and the current solution x and current best solution x_{best} (if necessary) are updated by x' . Iterations are repeated until all events are scheduled, and the obtained feasible timetable is returned.

4 Computational Experiments

4.1 Experimental Settings

We investigated the performance of the RPNS algorithm on the benchmark set of Socha *et al.* [6] because this benchmark set has been intensively tackled by many algorithms. This benchmark set consists of 11 instances, which are divided into three categories (5 small instances, 5 medium instances, and one large instance) according to the number of courses, rooms, and students. The numbers of (courses, rooms, students) are (100, 5, 80) for the small instances, (400, 10, 200) for the medium instances, and (400, 10, 400) for the large instance. We do not show results for the small instances because some of the previous approaches in the literature as well as RPNS can find feasible timetables with a penalty cost of zero, which makes these instances useless for comparison purposes.

The RPNS algorithm was implemented in C++ and was executed in a virtual machine environment (*i.e.*, each job is executed on a single core, but multiple jobs may be executed in the same node) on a cluster with Intel Xeon 2.93 GHz nodes. We performed the RPNS algorithm with various combinations of tabu tenure ($T = 0, 5, 10, 20, 30, 50, 70, 100, 120, 150$, and, 200) and neighborhood size ($ratio = 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.5$, and 1) in order to investigate their effects and relation on the performance. We set $iterMax = \lceil \frac{100,000}{ratio} \rceil$ in order to make the total number of evaluations of solutions in a single run (and therefore the computation time) the same regardless of the difference in the neighborhood size. For each of all configurations, we performed the RPNS algorithm 10 times on each instance.

4.2 Results

Figure 1 shows the results of the RPNS algorithm for all possible pairs of $ratio$ and T ; each curve, which corresponds to a value of $ratio$, is a plot of the average penalty values over 10 runs against the different values of T .

The RPNS algorithm with the full neighborhood ($ratio = 1$) finds high quality timetables that improves the lowest penalty timetables reported in the literature (except for instance **large**) if T is set to the best value for each instance. However, the quality is fairly sensitive to the value of T , meaning that the trade-off between exploitation and exploration must be controlled appropriately to achieve a high performance.

We can see that when T is set to the best value for each value of $ratio$, the use of partial neighborhoods ($ratio < 1$) improves the result of the full neighborhood ($ratio = 1$) unless the neighborhood size is too small. This is attributed to the trade-off between the positive effect of the increase in the number of iterations and the negative effect of the decrease in the possibility of finding a better solution in the neighborhood at each iteration. An important observation is that the best value of T decreases as the value of $ratio$ is decreased. The reason for this is that a reduction of the neighborhood size has an effect to diversify the search. An interesting observation is that the best performance over all possible pairs of $ratio$

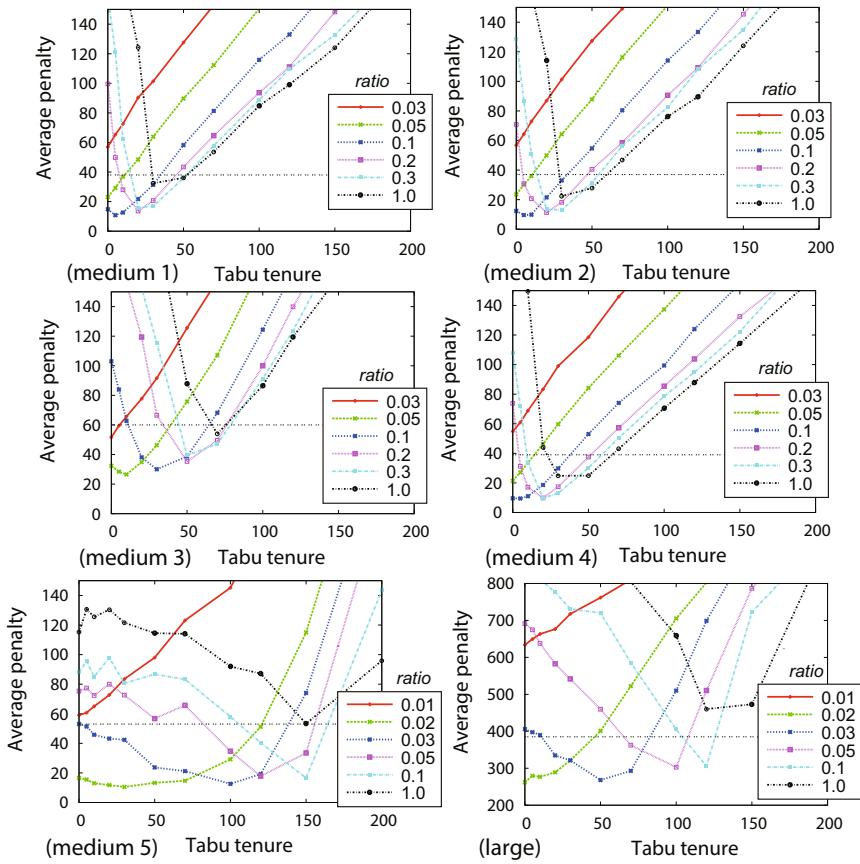


Fig. 1. Results of RPNS ($iterMax = \lceil \frac{100,000}{ratio} \rceil$), where horizontal dotted lines represent the best penalty values found in the literature

and T seems to be obtained when the neighborhood size is optimized under the setting of $T = 0$. For example, in the result on **medium 1**, the best performance is obtained by setting $T = 5$ and $ratio = 0.1$, but a better (or similar) result will be obtained by setting $T = 0$ and $ratio$ between 0.05 and 0.1. This is a little bit surprising because the strength of tabu search is completely spoiled in the setting of $T = 0$, meaning that a mechanism of diversifying the search depends solely on the reduction of the neighborhood size. From a practical standpoint, this is a nice property because the best performance is attained by adjusting only one parameter $ratio$ instead of adjusting both parameters T and $ratio$.

4.3 Analysis

In RPNS, the reduction of the neighborhood size plays an important role in diversifying the search, and we analyze this effect in more detail. Graph (a) of Figure 2 shows the number of students attending to each of the events for

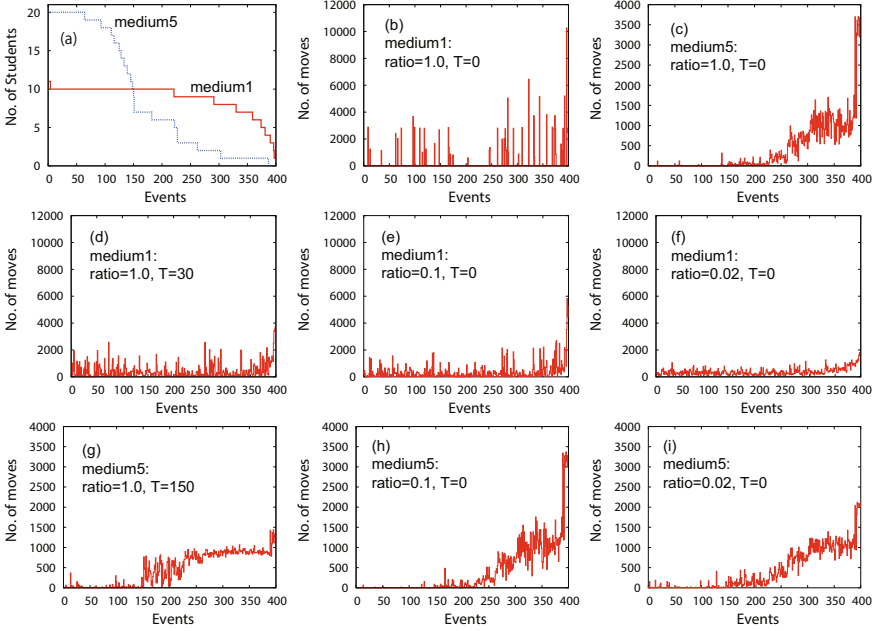


Fig. 2. (a) The number of students attending to each of the events, and (b–i) the cumulative number of moves for each of the events in a single run of RPNS

medium 1 and medium 5, where the events are re-indexed in descending order of the number of the students. We can see that the number of students varies widely among the events in medium 5, while the difference in the number of students is relatively small in medium 1. The other graphs (b)–(i) show the cumulative number of moves for each of the events (e.g. if events e_1 and e_2 are exchanged, both counts of e_1 and e_2 are incremented) in a single run of the RPNS algorithm for several settings of *ratio* and T on the two instances, where the events are re-indexed in the same order as in the graph (a). Note that the cumulative number of moves presented in Figure 2 is the result when the number of iterations is 100,000.

Graphs (b) and (c) show the results of a setting ($ratio = 1, T = 0$) on the two instances. We can see that in medium 5 the cumulative number of moves tends to increase as the number of students decreases (graph (c)). This is a natural consequence because the impact of an event on the penalty cost increases as the number of students increases. A similar situation does not occur in medium 1 because the difference in the number of students between the events is small, but *cycling* occurred frequently during the search and therefore the cumulative number of moves are concentrated in a part of the events (graph (b)).

Graphs (d)–(f) shows the results of three settings ($ratio = 1, T = 30$), ($ratio = 0.1, T = 0$), and ($ratio = 0.02, T = 0$) on medium 1, where $T = 30$ is the best value (when $ratio = 1$) and $ratio = 0.1$ is the best value

Table 1. Comparisons with the state-of-the-art algorithms

	GDTs	NGDHH	ENGdHH	HHSa	RPNS	
	(5 runs) best	(10 runs) best	(20 runs) best	(10 runs) best	(10 runs) ave.	(10 runs) best
medium 1	78	71	38	99	14.8	7
medium 2	92	82	37	73	12.4	10
medium 3	135	112	60	130	32.3	24
medium 4	75	55	39	105	9.6	5
medium 5	68	103	55	53	16.6	6
large	556	777	638	385	262.0	205

(when $T = 0$) for this instance. Graphs (g)–(i) shows the results of three settings ($ratio = 1$, $T = 150$), ($ratio = 0.1$, $T = 0$), and ($ratio = 0.02$, $T = 0$) on medium 5, where $T = 150$ is the best value (when $ratio = 1$) and $ratio = 0.02$ is the best value (when $T = 0$) for this instance.

Compared to the result of the full neighborhood ($ratio = 1$) with $T = 0$ (graphs (b) and (c)), the distribution of the cumulative number of moves is spread by setting $T = 30$ (medium 1) and $T = 150$ (medium 5) through the effect of tabu search. More importantly, we can see that a similar effect is obtained by decreasing the neighborhood size even if the value of T is set to zero, and this effect becomes more prominent as the value of $ratio$ is decreased. These results show that the degree of diversification of the search can be controlled by the neighborhood size.

4.4 Comparisons with Other Algorithms

Finally, we compare the performance of RPNS ($iterMax = \lceil \frac{100,000}{ratio} \rceil$) with those of leading algorithms that have shown competitive performance on the benchmark set of Socha *et al.* Table 1 shows the comparison results. The compared algorithms, which are selected from about thirty algorithms found in the literature are Great Deluge with Tabu Search (GDTs) [1], Non-linear Great Deluge Hyper Heuristic (NGDHH) [4], Extended version of Non-linear Great Deluge Hyper Heuristic (ENGdHH) [5], and Hybrid Harmony Search Algorithm (HHSa) [2]. For RPNS, we present the best and average results of 10 runs obtained with the setting of $T = 0$ and the best value of $ratio$ for each instance (see Figure 1) where the values of $ratio$ are 10 (medium 1), 10 (medium 2), 5 (medium 3), 10 (medium 4), 2 (medium 5), and 2 (large), respectively. For each of the compared algorithms, the best results obtained by multiple runs with various parameter settings (if experiments were conducted with various settings) are presented. Note that our purpose here is not to allege the superiority of RPNS over the compared algorithms because the value of $ratio$ was adjusted for each instance in RPNS, but we can see that the quality of the timetables obtained by RPNS is far ahead of others.

The average computation times for a single run of RPNS with the best values of $ratio$ (and $T = 0$) were 323 seconds (medium 1), 282 seconds (medium 2), 451 seconds (medium 3), 315 seconds (medium 4), 826 seconds (medium 5), and 418

seconds (large), respectively. For the compared algorithms, the average computation times for a single run for each of the instances were approximately 12 hours (GDTS), 3–5 hours (NGDHH), 2.5–5 hours (ENGDDH), and 6 hours (HSA), respectively. We can see that the computational effort of RPNS is reasonable even allowing for the differences in the computer speed and implementation.

5 Conclusion and Future Work

We have proposed an tabu search algorithm using an candidate list strategy with random sampling (random partial neighborhood search, RPNS) for the the university course timetabling problem, where the neighborhood size can be adjusted by a parameter *ratio*. Experimental results show that the degree of diversification of the search can be controlled by the neighborhood size, and the RPNS algorithm attains a very good performance when the neighborhood size is set properly especially when the tabu tenure is set to zero. In fact the quality of the timetables obtained by the RPNS algorithm with the best value of *ratio* is far ahead of those of the state-of-the-art algorithms. However, the quality is fairly sensitive to the value of *ratio*. At the current moment, it is difficult to estimate the best value of *ratio* for a given instance before or during the search, and a possible direction for future research is to develop a self-adapting mechanism for the value of *ratio*.

References

1. Abdullah, S., Shaker, K., McCollum, B., McMullan, P.: Construction of course timetables based on great deluge and tabu search. In: Proceedings of MIC 2009: VIII Metaheuristic International Conference, pp. 13–16 (2009)
2. Al-Betar, M.A., Khader, A.T., Zaman, M.: University course timetabling using a hybrid harmony search metaheuristic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 42(5), 664–681 (2012)
3. Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic Publishers (1997)
4. Obit, J., Landa-Silva, D., Ouelhadj, D., Sevaux, M.: Non-linear great deluge with learning mechanism for solving the course timetabling problem. In: 8th Metaheuristics International Conference, MIC 2009 (2009)
5. Obit, J.H., Landa-Silva, D., Sevaux, M., Ouelhadj, D.: Non-linear great deluge with reinforcement learning for university course timetabling. In: *Metaheuristics–Intelligent Decision Making*. Series Operations Research/Computer Science Interfaces. Springer (2011)
6. Socha, K., Knowles, J.D., Sampels, M.: A $\mathcal{M}\mathcal{A}\mathcal{X} - \mathcal{M}\mathcal{I}\mathcal{N}$ ant system for the university course timetabling problem. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) *Ant Algorithms 2002*. LNCS, vol. 2463, pp. 1–13. Springer, Heidelberg (2002)
7. Voudouris, C., Tsang, E.P., Alsheddy, A.: *Guided local search*. Springer (2010)

Balancing Bicycle Sharing Systems: An Analysis of Path Relinking and Recombination within a GRASP Hybrid*

Petrina Papazek, Christian Kloimüller, Bin Hu, and Günther R. Raidl

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/1861, 1040 Vienna, Austria
{papazek,kloimueller,hu|raidl}@ads.tuwien.ac.at

Abstract. In bike sharing systems, a vehicle fleet rebalances the system by continuously moving bikes among stations in order to avoid rental stations to run entirely empty or full. We address the static problem variant assuming initial fill levels for each station and seek vehicle tours with corresponding loading instructions to reach given target fill levels as far as possible. Our primary objective is to minimize the absolute deviation between target and final fill levels for all rental stations. Building upon a previously suggested GRASP hybrid, we investigate different approaches for hybridizing them with Path Relinking (PR) and simpler recombination operators. Computational tests on benchmark instances derived from a real world scenario in Vienna give insight on the impacts of the PR and recombination techniques and manifest that certain PR extension improve the results significantly. Ultimately, a hybrid exclusively searching a partial PR path in the neighborhood of the guiding solutions turns out to be most fruitful.

1 Introduction

Public Bicycle Sharing Systems (BSSs) are booming in many cities around the globe. A BSS comprises a set of automated rental stations, which allows users to rent and return bikes at any station of the system at any time. Establishing BSSs in cities is beneficial as they augment public transport very well, are “green” alternatives to motorized traffic, and contribute to public health [1]. However, BSSs face one major issue: without actively redistributing bikes among stations, i.e. balancing the stations, most would either run out of available bikes or free slots. This natural disbalance of a BSS originates from diverse circumstances, such as topography, commuting patterns, and weather conditions. To keep the system in balance, most BSS operators actively redistribute bikes among stations by a vehicle fleet.

* This work is supported by the Austrian Research Promotion Agency (FFG), contract 831740. The authors thank Citybike Wien, the Austrian Institute of Technology (AIT), and Energie und Umweltagentur Niederösterreich (eNu) for the collaboration in this project.

We consider the *Balancing Bicycle Sharing System* (BBSS) problem as introduced in [2]: Given a set of vehicles and a certain time budget, our goal is to find vehicle tours with corresponding bicycle loading instructions such that the stations' fill levels are brought to specified target values as far as possible.

Building upon our previous work, in particular a GRASP hybrid, we investigate different extensions by Path Relinking (PR) and simpler recombination operators and their impacts. While straight-forward applications of PR do not yield significant improvements in solution quality but rather increase the computational costs substantially, a variant, where only a part of the complete PR path in the vicinity of the guiding solution is searched, turns out to be fruitful. The GRASP hybrid in conjunction with this restricted form of PR is able to improve previously leading results on mid-size instances significantly.

2 Related Work

Although BBSS is a relatively new problem domain already several different algorithmic approaches have been suggested for diverse variants of it. Most of them apply integer linear programming techniques [3,4,5,6], whose applicability, however, is restricted to small instances or strongly simplified problem variations.

In our previous work we have developed a greedy construction heuristic which we improved by the PILOT method [7]. As metaheuristic approaches we established a Variable Neighborhood Search (VNS) [2] with an embedded Variable Neighborhood Descent (VND) and GRASP [7]. Our results about the static case have been refined together with comprehensive computational tests in the *Journal of Global Optimization* [8]. A more detailed description of these approaches can be found in Section 4. Comparing different work on BBSS is usually difficult because they consider various specific problem variants. Di Gaspero et al. [9,10] proposed approaches based on constraint programming, ant colony optimization, and large neighborhood search and tested them on the same benchmark instances as ours, though they could not outperform our previous approaches mentioned above.

For the dynamic case, where the system is rebalanced while usage is simulated, only few work has been published so far [11]. In a recent work we considered this scenario and extended our techniques to handle expected dynamic user demands without relying on an expensive time-discretization [12].

Much more work exists on other aspects of BSSs, such as an optimal network design [13], system characteristics and usage patterns [14], but they are not in the scope of this paper.

3 Problem Definition

We consider the static or offline BBSS problem variant which neglects user interaction during rebalancing and model the BSS as a complete directed graph $G_0 = (V_0, A_0)$. The vertices V_0 comprise all rental stations V and the depot 0 , which is the start and end point of the tours. The arcs A_0 connect all vertices

and are weighted with time $t_{u,v}$. This time represents the travel time between nodes $u, v \in V_0$ including an average service time for loading or unloading bikes at node v . Furthermore, we are given for each station $v \in V$ the capacity C_v , i.e., the number of bike slots, the initial fill level p_v , and a target fill level q_v . The BSS operator employs a fleet of vehicles L for distributing the bikes. Each vehicle $l \in L$ starts empty at the depot and may *visit* an arbitrary number of stations before it has to return *empty* to the depot again. Each vehicle $l \in L$ has associated a bike capacity Z_l .

A solution to the BBSS problem comprises a tour for each vehicle $l \in L$ and corresponding loading instructions for each stop. We define this ordered sequence of visited stations as $r_l = (r_l^1, \dots, r_l^{\rho_l})$ with $r_l^i \in V$, $i = 1, \dots, \rho_l$, and ρ_l being the number of visited stations. An important aspect is that stations may be visited multiple times by the same or different vehicles. Each visit has associated loading instructions $y_l^i \in \{-Z_l, \dots, Z_l\}$ with $l \in L$ and $i = 1, \dots, \rho_l$, specifying how many bikes are to be picked up ($y_l^i > 0$) or delivered ($y_l^i < 0$) at that visit. A solution is feasible if vehicle and station capacities are never exceeded, the number of bikes available at stations is never below zero, a vehicle does not deliver more bikes than it has actually loaded, and if the working time t_l of a vehicle $l \in L$ does not exceed a given total time budget \hat{t} .

Let a_v be the final number of bikes at each station $v \in V$ after rebalancing. The primary objective is to minimize the deviation from the target values q_v , i.e., the disbalance $|a_v - q_v|$ at each station $v \in V$. As secondary objectives we want to minimize the total number of loading operations $|y_l^i|$ over all visits ρ_l and all vehicles L and the working time t_l of all drivers $l \in L$. This is expressed by the following objective function:

$$\min \underbrace{\omega^{\text{bal}} \sum_{v \in V} |a_v - q_v|}_{\text{disbalance}} + \underbrace{\omega^{\text{load}} \sum_{l \in L} \sum_{i=1}^{\rho_l} |y_l^i|}_{\text{loading operations}} + \underbrace{\omega^{\text{work}} \sum_{l \in L} t_l}_{\text{working time}} \quad (1)$$

The scaling factors ω^{bal} , ω^{load} , and ω^{work} control the relative importance of these terms. We assume that any improvement in balance is always preferred over decreasing the number of loading actions or reducing the working time and set the scaling factors accordingly ($\omega^{\text{bal}} = 1$ and $\omega^{\text{load}} = \omega^{\text{work}} = 1/100\,000$). The reason is that a balanced system is top priority for maximizing customer-satisfaction while from the operator’s point of view, workers are paid for the whole shift length anyway, and therefore a reduction in the tour lengths is just a secondary aspect.

4 Metaheuristics Approaches

This section gives an overview on the metaheuristic approaches we proposed in [8]. All of these approaches utilize an incomplete solution representation by considering vehicle tours only; corresponding loading operations are computed via an auxiliary procedure whenever a solution is evaluated. From the different

variants for this procedure considered in [8], we use here the fastest greedy heuristic [2,8], which is the most reasonable approach in practice since it scales best for large instances and nevertheless yields close to optimal loading operations.

To create an initial solution we employ two alternative construction heuristics: a greedy construction heuristic (GCH) and an extension of it based on the PILOT method [15]. GCH sequentially constructs vehicle tours following a local best successor strategy. To derive a tour for each vehicle, we compute the maximum number of bicycles γ_v that can be picked up or delivered at any not yet balanced station v without exceeding/deceeding its target value. For each vehicle, we construct a tour by starting at the depot and evaluating the next best successor by the ratio $\gamma_v/t_{u,v}$. Here, we only consider a station v if enough time remains to return to the depot afterwards. If there are no further feasible stations left, we proceed with the next vehicle. Afterwards, we derive loading instructions by an auxiliary heuristic. The PILOT construction heuristic (PCH) extends the greedy construction heuristic by overcoming possibly shortsighted successors. In particular, PCH evaluates each potential successor more accurately by constructing a complete temporary route utilizing the objective function value as evaluation criterion. This approach requires more time than GCH, but yields substantially better results in return.

For locally improving candidate solutions, we employ a Variable Neighborhood Descent (VND) [16] using seven neighborhood structures and applying a best improvement strategy [8]: remove station, insert unbalanced station, replace station, intra-or-opt, 2-opt* inter-route exchange, and intra-route 3-opt.

As the solution construction followed by VND is still quite fast and improvement potential remains, the approach is further extended into a Variable Neighborhood Search (VNS) [8] with the following shaking operators: move-sequence, exchange-sequence, destroy-&-recreate, and remove-stations.

In addition to the VNS, we investigated a hybrid Greedy Randomized Adaptive Search Procedure (GRASP) [17] by iteratively applying randomized versions of either GCH or PCH, locally improving each solution with the VND, and finally returning the overall best solution. Here, the PCH version with a random neighborhood in the VND turned out to perform best.

In the experimental evaluation in [8], VNS yields the best results on small and mid-size instances with up to 300 stations, while PCH-GRASP performed better on large instances with up to 700 stations.

5 Path Relinking and Recombination in GRASP

Glover et al. [18] define PR as the evolutionary technique of altering an initial solution I towards a guiding solution G – typically a solution from an elite set – by a series of simple moves. These moves describe a trajectory or path of (not necessarily feasible) solutions, and a best encountered solution is returned as result. According to numerous studies such as [19,20], PR yields promising results on diverse related vehicle routing problems. Moreover, simpler recombination techniques as they are mainly used in evolutionary algorithms are able to yield

possibly promising candidate solutions from the joined properties of two input solutions in a fast manner. Therefore, we investigate extensions of the above hybrid GRASP by PR and simpler recombination techniques. This section describes several specific approaches in detail, which we found most meaningful in the context of BBSS, and explains how to embed it into GRASP.

5.1 PR and Recombination Variants

In order to iteratively transform solution I into solution G inside PR, we need to define basic moves. Again, we consider loading instructions to be always calculated by the auxiliary greedy heuristic for each intermediate candidate solution on the fly and thus, solely concentrate on the tours. One basic move edits a single vehicle tour r_l by removing, adding, or replacing exactly one station.

We adopt the common principle from PR for other vehicle routing problems to match each tour of I with a tour in G and individually relink or recombine each pair of corresponding tours. As we consider the case of having a not necessarily homogeneous vehicle fleet, we relate each vehicle tour of I with the tour of the same vehicle in G . Then, we transform the tours from I into corresponding ones from G , yielding a series of intermediate solutions. These intermediate solutions are not necessarily feasible as the relinked tours may exceed the allowed time budget \hat{t} . Consequently, we repair such infeasible tours by pruning them from the end before they are evaluated. In the following we denote by $r_l(G)$ the tour of vehicle l in solution G and by $r_l(I)$ the corresponding tour in solution I , $l \in L$; $\rho_l(G)$ and $\rho_l(I)$ denote their respective tour lengths. We consider the following operators for systematically generating intermediate solutions.

Sequential Replace PR (Seq-PR). Within this basic PR operator, we sequentially transform each tour $r_l(I)$, $l \in L$ step-by-step with basic moves into G 's corresponding tour. This is achieved by iterating over the stops r_l^i with $i = 1, \dots, \rho_l(G)$ and replacing each corresponding stop $r_l^i(I)$ by $r_l^i(G)$ or adding it if I 's original tour was shorter.

One-Point-Recombination (OP-Rec). Starting from the parent solutions I and G , we randomly select a vehicle $l \in L$ and a crossover position $p \in \{1, \dots, \min(\rho_l(I), \rho_l(G))\}$. In a first offspring, the tour r_l becomes

$$(r_l^1(I), \dots, r_l^{p-1}(I), r_l^p(G), \dots, r_l^{\rho_l(G)}(G))$$

and in a second

$$(r_l^1(G), \dots, r_l^{p-1}(G), r_l^p(I), \dots, r_l^{\rho_l(I)}(I)).$$

For each of the next vehicles $l \in L$ we randomly decide whether its tour is copied from $r_l(G)$ or $r_l(I)$ without any further change.

Single Tour Recombination (ST-Rec). This recombination variant basically works like OP-Rec by first selecting a vehicle and performing one-point crossover on the two corresponding tours, but then, all remaining tours are only adopted from I for the first offspring and only G for the second offspring, respectively. Thus, this operator changes a parent solution only in a part of one tour.

Restricted Multistart PR (MS-PR). This operator combines concepts from Seq-PR and ST-Rec. A main drawback of Seq-PR is its high computational cost. To speed up Seq-PR, we skip the evaluation of solutions in the middle part of the tours, since most potential solutions are in the close neighborhood of the optimized tours in I and G . As we found out in preliminary analysis, the best solutions on trajectories from I to G are almost always located relatively close to G and for this reason we focus on the final stations of tours. We therefore restrict ourself in MS-PR to *final parts* of paths from I to G . This is achieved by starting with a whole copy of G and replacing just one tour $r_l(G)$, $l \in L$, by a copy from $r_l(I)$. This tour $r_l(I)$ is then step-by-step relinked as in Seq-PR until G is reached again. In particular, it turned out most successful to solely evaluate the first and three final stations of the tour. If there is more than one vehicle, we perform the same procedure also for all the other vehicles in L ; i.e., we perform multiple starts, one for each vehicle, which gives the operator its name.

Applying VND to the final best solution from the PR operator as suggested by [21] typically improves our results, because of frequent unfavorable ordering of stations by merely merging arbitrary tours. Note that our PR primarily uses replace operations without more sophisticated best improvement insert strategies. To integrate a reordering of stations into the PR or recombination operator, we can additionally employ VND to each intermediate/offspring solution.

However, performing the full VND at each step may soon become too expensive for large instances. In order to speed up the process, we use a limited variant of VND, which works as follows: For each intermediate PR solution, we test if applying a single move of each VND neighborhood leads to a solution that is better than I . Only if this is the case, a full VND is applied. This way lots of mediocre solutions on the path are skipped relatively quickly.

5.2 Embedding in GRASP

For applying PR and recombination operators within our PILOT-GRASP from [8], we add a memory to GRASP through an elite set of best solutions. Fiesta et al. [21] proposed different adaptive variants to join PR and GRASP, in particular Path Relinking GRASP (PR-GRASP), which we adopt here.

In PR-GRASP we employ the PR or recombination operator on each new VND improved GRASP solution and a randomly selected member of the elite set. The elite set is initialized by adding solutions from the GRASP procedure with the condition that they must be different from each other. If the initial elite set is sufficiently large (e.g., after five GRASP iterations), we employ the PR or recombination operator in each GRASP iteration. Before embarking the next

GRASP iteration, we consider to incorporate the PR or recombination enhanced candidate solution to the elite set in case it is different from all other already included members. In this context, a pure random update strategy turned out to perform best: We select a random element from the elite set and replace it by the new candidate solution if the objective value of the new one is better. In order to enhance diversity and quality in the elite set, we also tested the common approach of considering a minimal solution edit distance as proposed by [21]. However, preliminary tests revealed that this distance-based replacement strategy did not yield significantly better solutions but introduced a non-negligible runtime overhead due to the distance calculations.

6 Computational Analysis

In this section we show the results of our PR and recombination hybrids and compare them with the leading approaches from [8]. The instances of our benchmark sets¹ have been derived from the real-world scenario in Vienna provided by Citybike Wien and the AIT. The initial fill levels p_v of the stations are taken from a historical snapshot of the system in Vienna whereas the target fill levels q_v are calculated according to the estimated user demands at the particular stations. For further reference on the computation of target values, please, refer to our publication in the *Journal of Global Optimization* [8]. We consider an amount of stations $|V| \in \{60, 90, 180, 300, 400, 500\}$, a time budget \hat{t} of 2, 4, 8 hours, and a number of vehicles $|L|$ between 1% and 2% of $|V|$ in order to test reasonable settings for practical scenarios. Large instances beyond 500 stations are neglected because the PR-GRASP variants are able to run only a few iterations, and PR extensions would make no sense in such situations. All benchmark sets include 30 instances and represent unique combinations of $|V|$ and $|L|$. We run each instance on a single core of an Intel Xeon E5540 machine with 2.53 GHz. For every algorithmic variant we use a common CPU time limit of 15 minutes for small instances with 30 stations, 30 minutes for medium instances with 60 to 90 stations, and 60 minutes for large instances with more than 90 stations.

Tables 1 and 2 include the results for the basic PR and recombination variants in conjunction with PILOT-GRASP as well as our previously developed metaheuristics from [8] and our most successful PR-GRASP variants: PILOT-GRASP with ST-Rec including full VND and MS-PR including limited VND. We list the mean objective values \overline{obj} , the counted best values $\#best$ (i.e., the number of instances where the approach has been superior to all other variants over both tables), and the number of major GRASP-iterations $\overline{g_{tot}}$ for each instance.

The OP-Rec operator is comparatively fast and enables many GRASP iterations. In particular, it requires only two VND evaluations per iteration and is thereby substantially faster than the Seq-PR operator. Both variants, Seq-PR as well as OP-Rec, are only competitive on smaller instances and are worse for mid-size and large instances. Thus, it is most interesting to concentrate on instances of smaller size for these PR variants. We observe that for these smaller

¹ https://www.ads.tuwien.ac.at/w/Research/Problem_Instances#bss

Table 1. Computational results of Seq-PR- and OP-Rec-GRASP

Inst. set $ V $ $ L $ \hat{t}	Seq-PR-GRASP						OP-Rec-GRASP					
	full VND			limited VND			full VND			limited VND		
	#best	\overline{obj}	$\overline{g_{tot}}$	#best	\overline{obj}	$\overline{g_{tot}}$	#best	\overline{obj}	$\overline{g_{tot}}$	#best	\overline{obj}	$\overline{g_{tot}}$
30 1 2	26	147.33499	715197.0	26	147.33499	713813.0	26	147.33499	713312.0	26	147.33499	723585.5
30 1 4	22	95.60334	43876.5	28	95.40335	20617.0	28	95.40335	53717.0	22	95.73667	74951.0
30 1 8	24	29.20639	3605.5	29	29.20639	1142.0	28	29.27305	5352.5	26	29.20639	8773.0
60 1 4	18	270.93692	44641.5	26	269.93695	19738.0	26	269.93695	48428.5	20	270.73692	65299.5
60 1 8	15	170.80700	4086.0	20	170.34035	1273.0	19	170.27369	5616.0	15	170.87366	9330.5
60 2 2	24	293.93664	92158.0	26	293.80331	45680.5	26	293.80331	116599.5	24	293.93664	133458.0
90 2 4	8	347.60720	6505.0	12	346.27390	1783.5	19	345.94057	8604.0	5	347.80719	14387.5
90 2 8	5	174.94703	647.0	0	176.28033	99.0	0	175.88033	1039.5	0	174.94702	1971.0
90 4 4	0	197.94672	1537.0	0	198.54669	217.5	0	197.61337	2490.0	1	197.21339	5547.5
180 4 4	1	721.21426	1459.0	3	721.14757	236.0	2	721.34759	2486.5	1	722.01424	4505.0
180 4 8	0	380.29379	146.5	0	385.09377	16.0	0	383.82704	261.0	1	379.89377	654.5
180 5 8	0	271.23286	91.0	0	275.83275	11.0	0	275.56617	160.5	0	269.69964	465.5
300 6 4	0	1249.08822	311.5	0	1253.88816	38.0	0	1253.22150	530.0	0	1249.08824	1220.5
300 6 8	0	718.77440	38.0	0	725.04097	7.0	0	724.44097	60.5	3	716.24113	187.0
300 9 8	0	399.45832	18.0	0	403.59148	6.0	0	402.99165	29.0	0	396.45840	103.0
400 8 4	0	1660.96188	128.5	0	1668.42848	16.0	0	1665.62856	237.5	0	1660.36191	563.0
400 8 8	0	954.52126	17.0	0	956.92125	6.0	0	957.92127	27.0	3	950.72133	86.5
400 12 8	2	530.27760	10.0	0	531.27760	6.0	0	532.07758	14.0	8	525.81105	49.5
500 10 4	1	2094.23569	61.0	0	2104.63555	9.0	0	2101.96892	110.5	1	2092.36906	297.0
500 10 8	1	1207.40169	11.0	2	1211.06830	6.0	0	1211.80159	15.5	3	1203.53505	47.0
500 15 8	1	665.83047	7.0	2	666.96383	6.0	2	667.16383	9.0	3	662.96395	27.0
Total	148	12581.61667	914552.0	174	12631.01597	804726.5	176	12623.41628	959100.0	162	12556.95064	1045509.0

Table 2. Computational results of ST-Rec- and MS-PR-GRASP

Inst. set $ V $ $ L $ \hat{t}	VNS			PILOT-GRASP			ST-Rec-GRASP			MS-PR-GRASP		
	#best	\overline{obj}	$\overline{g_{tot}}$	#best	\overline{obj}	$\overline{g_{tot}}$	#best	\overline{obj}	$\overline{g_{tot}}$	#best	\overline{obj}	$\overline{g_{tot}}$
30 1 2	28	147.20166	1215760.5	29	147.13500	618965.5	26	147.33499	724028.0	26	147.33499	698800.0
30 1 4	27	95.20336	269610.5	23	95.73667	100840.0	28	95.40335	52850.5	22	95.80334	60057.5
30 1 8	26	29.20639	37254.0	26	29.27305	10879.0	28	29.20639	5422.0	25	29.27305	6728.0
60 1 4	29	269.60362	310747.5	17	271.20358	77673.0	26	269.93695	48277.0	18	270.93692	55162.0
60 1 8	18	170.47367	44431.5	12	171.00699	10889.5	25	170.20702	5539.5	13	170.87366	7755.0
60 2 2	23	293.80331	505746.0	22	294.06999	190196.5	27	293.60332	118511.5	23	294.00330	79788.0
90 2 4	15	346.27390	50771.5	2	348.74052	17740.5	21	345.87391	8492.5	5	347.40721	9292.5
90 2 8	18	173.14705	7677.0	2	175.34702	2325.5	2	176.08031	1015.0	3	174.74701	1296.5
90 4 4	21	193.74679	15854.5	0	197.88006	6829.5	5	196.41340	2693.5	5	195.41342	2103.5
180 4 4	15	718.08096	14634.5	1	722.61423	5184.0	2	720.21427	2739.0	6	719.61426	2174.5
180 4 8	12	374.22726	1687.5	0	379.96048	739.0	0	382.42709	278.0	17	372.76058	308.5
180 5 8	11	264.03307	996.5	0	269.83290	510.0	0	272.96620	187.5	19	263.16634	187.5
300 6 4	19	1241.28831	3209.0	1	1248.82158	1355.5	0	1248.95489	699.0	10	1242.35494	459.0
300 6 8	6	715.10780	322.5	0	715.64113	201.0	0	722.04101	75.0	22	709.37453	79.5
300 9 8	3	403.25824	127.5	9	394.39177	114.5	0	401.52495	39.5	18	391.99176	37.0
400 8 4	13	1654.16199	1179.5	2	1658.49534	612.0	2	1660.89527	336.0	13	1653.16199	196.0
400 8 8	0	958.58794	120.0	6	949.05467	91.5	0	956.32119	37.0	21	946.52135	37.0
400 12 8	0	543.47730	53.0	13	524.34443	52.0	1	531.27757	21.0	7	526.47768	19.0
500 10 4	5	2092.16902	517.0	2	2091.90240	322.0	0	2095.03570	177.5	21	2084.03580	90.0
500 10 8	0	1225.46808	55.5	11	1202.93505	49.5	1	1209.66824	21.5	12	1202.66836	21.0
500 15 8	0	690.22984	26.5	16	660.43057	28.0	3	666.63055	13.0	4	664.96383	12.0
Total	289	12598.74956	2480782	194	12548.81743	1045598	197	12592.01657	971453.5	310	12502.88432	924604.0

instances it does not make sense to apply the limited VND since only a few solutions are actually improved. Moreover, ST-Rec yields significantly better results than the original OP-Rec operator because we destroy less well working parts of the tours.

Comparing with state-of-the-art approaches (i.e., VNS and PILOT-GRASP), the PR extensions are able to boost the performance of GRASP on medium instances which have so far been dominated by VNS. Among the PR-GRASP variants, we observe that MS-PR-GRASP is superior on medium and large instances with 180 to 500 stations whereas ST-Rec-GRASP is able to enhance GRASP for small instances with 60 to 90 stations. While VNS still performs

best on medium instances with 90 stations in overall, the results indicate that our new GRASP variants are able to catch up on instances with smaller time budgets of 2 and 4 hours. These observations are confirmed by the Wilcoxon Rank Sum test on an error level of 5%.

7 Conclusions and Future Work

To improve results of a state-of-the-art PILOT-GRASP hybrid for the static BBSS variant, we analyzed different variants of PR and recombination operators to obtain new promising candidate solutions by joining parts of two parental solutions. Seq-PR follows a very traditional way of performing PR and introduces a quite high computational overhead. In contrast, OP-Rec is an implementation of a fast, rather classical one-point crossover in the context of our solution representation. It turned out that intermediate solutions that are constructed from large parts of both parent solutions are in general less promising than when most properties are inherited from one parent and only a smaller portion is adopted from the second parent. Consequently, we came up with ST-Rec, which performs the crossover only on a single tour and adopts all other tours from a single parent. It turned out that this generally less disruptive recombination yields significantly better results. We then also modified Seq-PR into MS-PR, where we investigate only parts of whole paths from an initial solution to the guiding solution that are close to the guiding solution and skip solutions in the middle of tours. This modification was most fruitful and yields the best results. In this way we could obtain new leading solutions and significantly improved average results especially on larger instances with up to 500 nodes.

More generally, we strongly believe that also in other problems, the parts of path relinking trajectories close to either the initial or the guiding solutions may be more promising than the middle ones, and consequently, focusing the search on those ends may yield significant improvements or speedups. With respect to BBSS, we intend to integrate PR also in our GRASP-variant for the dynamic BBSS [12], where user interactions during rebalancing are also considered.

References

1. DeMaio, P.: Bike-sharing: History, impacts, models of provision, and future. *Public Transportation* 12(4), 41–56 (2009)
2. Rainer-Harbach, M., Papazek, P., Hu, B., Raidl, G.R.: Balancing bicycle sharing systems: A variable neighborhood search approach. In: Middendorf, M., Blum, C. (eds.) *EvoCOP 2013*. LNCS, vol. 7832, pp. 121–132. Springer, Heidelberg (2013)
3. Chemla, D., Meunier, F., Calvo, R.W.: Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization* 10(2), 120–146 (2013)
4. Raviv, T., Tzur, M., Forma, I.A.: Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transp. and Log.*, 1–43 (2013)
5. Benchimol, M., Benchimol, P., Chappert, B., De la Taille, A., Laroche, F., Meunier, F., Robinet, L.: Balancing the stations of a self service bike hire system. *RAIRO – Operations Research* 45(1), 37–61 (2011)

6. Schuijbroek, J., Hampshire, R., van Hoes, W.J.: Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems. Technical Report 2013-E1, Tepper School of Business, Carnegie Mellon University (2013)
7. Papazek, P., Raidl, G.R., Rainer-Harbach, M., Hu, B.: A PILOT/VND/GRASP hybrid for the static balancing of public bicycle sharing systems. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST. LNCS, vol. 8111, pp. 372–379. Springer, Heidelberg (2013)
8. Rainer-Harbach, M., Papazek, P., Hu, B., Raidl, G.R.: PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization* (2013), doi:10.1007/s10898-014-0147-5
9. Di Gaspero, L., Rendl, A., Urli, T.: A hybrid ACO+CP for balancing bicycle sharing systems. In: Blesa, M.J., Blum, C., Festa, P., Roli, A., Sampels, M. (eds.) HM 2013. LNCS, vol. 7919, pp. 198–212. Springer, Heidelberg (2013)
10. Di Gaspero, L., Rendl, A., Urli, T.: Constraint-based approaches for balancing bike sharing systems. In: Schulte, C. (ed.) CP 2013. LNCS, vol. 8124, pp. 758–773. Springer, Heidelberg (2013)
11. Contardo, C., Morency, C., Rousseau, L.M.: Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, Montreal, Canada (2012)
12. Kloimüller, C., Papazek, P., Hu, B., Raidl, G.R.: Balancing bicycle sharing systems: An approach for the dynamic case. In: *Evolutionary Computation in Combinatorial Optimization*, 12 p. (to appear, 2014)
13. Lin, J.R., Yang, T.H., Chang, Y.C.: A hub location inventory model for bicycle sharing system design: Formulation and solution. *Computers & Industrial Engineering* 65(1), 77–86 (2013)
14. Nair, R., Miller-Hooks, E., Hampshire, R.C., Bušić, A.: Large-scale vehicle sharing systems: Analysis of Vélip'. *Int. Journal of Sustain. Transp.* 7(1), 85–106 (2013)
15. Voß, S., Fink, A., Duin, C.: Looking ahead with the PILOT method. *Annals of Operations Research* 136, 285–302 (2005)
16. Mladenović, N., Hansen, P.: Variable neighborhood search. *Computers and Operations Research* 24(11), 1097–1100 (1997)
17. Resende, M., Ribeiro, C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) *Handbook of Metaheuristics*, pp. 219–249. Kluwer Academic Publishers (2003)
18. Glover, F., Laguna, M., Marti, R.: Fundamentals of scatter search and path relinking. *Control and Cybernetics* 29(3), 653–684 (2000)
19. Ho, S.C., Gendreau, M.: Path relinking for the vehicle routing problem. *Heuristics* 12(1-2), 55–72 (2006)
20. Rahimi-Vahed, A., Crainic, T., Gendreau, M., Rei, W.: A path relinking algorithm for a multi-depot periodic vehicle routing problem. *Heuristics* 19(3), 497–524 (2013)
21. Festa, P., Resende, M.G.C.: Hybridizations of GRASP with path-relinking. In: Talbi, E.-G. (ed.) *Hybrid Metaheuristics*. SCI, vol. 434, pp. 139–159. Springer, Heidelberg (2013)

Multiobjective Selection of Input Sensors for SVR Applied to Road Traffic Prediction

Jiri Petrlik, Otto Fucik, and Lukas Sekanina

Brno University of Technology, Faculty of Information Technology, IT4I Centre,
Czech Republic
{ipetrlik,fucik,sekanina}@fit.vutbr.cz

Abstract. Modern traffic sensors can measure various road traffic variables such as the traffic flow and average speed. However, some measurements can lead to incorrect data which cannot further be used in subsequent processing tasks such as traffic prediction or intelligent control. In this paper, we propose a method selecting a subset of input sensors for a support vector regression (SVR) model which is used for traffic prediction. The method is based on a multimodal and multiobjective NSGA-II algorithm. The multiobjective approach allowed us to find a good trade-off between the prediction error and the number of sensors in real-world situations when many traffic data measurements are unavailable.

Keywords: Road traffic forecasting, multiobjective feature selection, multiobjective genetic algorithms.

1 Introduction

Modern traffic sensors (induction loop detectors, radars and camera detectors) can measure various road traffic variables such as the *traffic flow* representing the number of vehicles passing a given road segment per time interval, the *occupancy* which is a dimensionless traffic variable describing the fraction of a time interval in which the current place is occupied by a vehicle, and the *arithmetic mean speed* of vehicles passing the current place. The detectors usually aggregate these data from intervals between 20 s and 5 min [1]. The information provided by traffic sensors is used in modern intelligent traffic systems (ITS), for traffic system planning and other purposes.

As the traffic sensors are not one-hundred percent reliable, the problem of estimation of missing values was identified. Various solutions to this problem have been proposed by means of modern soft computing methods which, in addition to the estimation, can also be employed to predict the future values on desired sensors. The predicted values can be utilized in ITS to control, for example, traffic lights and variable message signs. One of the most promising machine learning methods of short-term traffic flow forecasting is *support vector regression* (SVR). Previous methods based on SVR have not considered the selection of proper inputs (sensors). However, a proper selection of these sensors can significantly influence the quality of prediction.

In this paper, we propose a new multiobjective optimization method based on a genetic algorithm for selection of a subset of inputs for SVR. The proposed solution can be used for short-term traffic forecasting or for estimation of unmeasured values from broken sensors. Dealing with the missing values is important, because if the value from an input sensor is unavailable, the SVR method does not work at all! The proposed method is constructed as multiobjective because there is a natural trade-off between the error of prediction, the number of input sensors for SVR model and the data unavailability rate (a time fraction in which the SVR model can not be used because of missing data). The proposed method is evaluated using publicly available data and compared with a single objective optimization scenario.

The rest of the paper is organized as follows. Section 2 introduces the short-term traffic forecasting problem and methods based on SVR which can be used to solve this problem. Section 3 deals with multiobjective evolutionary algorithms. In Section 4, the proposed method is described. Experimental evaluation is performed in Section 5. Section 6 concludes the paper.

2 Road Traffic Forecasting Using SVR

Artificial neural networks and SVR were applied to solve the road traffic forecasting problem [2,4,5,6]. However, SVR becomes a more popular method in this task. SVR is a variant of support vector machine (SVM). While SVMs are usually used for classification problems, SVR is designed for regression and prediction problems [3]. The original SVM algorithm could work only as a linear classifier. In order to deal with non-linear problems, SVM/SVR was extended to support nonlinear kernel functions such as polynomial kernels, Gaussian radial basis kernels, and hyperbolic tangent kernels.

A SVR modification called Online-SVR (OL-SVR) [4] was previously used for short-term prediction of traffic behavior. The data from seven randomly selected highways were used to evaluate this method under typical and atypical traffic conditions. The atypical traffic conditions can appear, for example, during holidays or traffic incidents. The method was compared with neural networks (multilayer perceptron), Gaussian maximum likelihood (GML) and Holt's exponential smoothing. For typical traffic conditions the OL-SVR model outperformed the multilayer perceptron and Holt's exponential smoothing, but GML model provided better results. For atypical traffic conditions the OL-SVR model outperformed all other methods [4].

SVR requires to correctly set various meta-parameters such as the kernel type and regularization parameter. For example, chaotic simulated annealing was successfully used for parameter tuning. The results showed that SVR with optimized meta-parameters is as good as other techniques like seasonal autoregressive integrated moving average (SARIMA), seasonal Holt-Winter's model and back-propagation neural networks [5]. In another approach, a modified version of a particle swarm optimization (PSO) was utilized to find the optimal settings of SVR meta-parameters. The results proved that SVR model with

meta-parameters set by PSO can outperform the back propagation neural networks and ARMA model [6].

However, all the methods assumed that all sensors work nearly all the time, which is an unrealistic assumption. Hence the main objective of this paper is finding a solution which will work with unreliable sensors, i.e. missing data.

3 Multiobjective Genetic Algorithms

The most important objective in the prediction tasks is minimizing the error of prediction which is usually calculated by some error metrics such as the root mean squared error (RMSE). However, in real-world scenarios, other objectives have to be considered, for example, the number of data streams (sensors) has to be minimized because of their cost, maintainability, and reliability. In the context of this paper, the goal of the multiobjective scenario is to find the smallest subset of input sensors for which the number of missing values is minimal and the RMSE is minimal. In general, the multiobjective optimization problem can be defined in the following form:

$$\begin{aligned} & \text{minimize: } f_m(\mathbf{x}), & m = 1, 2, \dots, M \\ & \text{subject to: } g_j(\mathbf{x}) \geq 0 & j = 1, 2, \dots, J \\ & h_k(\mathbf{x}) = 0 & k = 1, 2, \dots, K \end{aligned} \quad (1)$$

where $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is a vector representing the solution consisting of n decision variables. The objective functions are denoted f_1, \dots, f_M . These functions are to be minimized. Functions $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ define the optimization constraints. In order to compare two solutions, Pareto-dominance relations were established [7]: Solution $\mathbf{x}^{(1)}$ dominates another solution $\mathbf{x}^{(2)}$ if the following conditions are satisfied: (1) The solution $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all objectives. (2) The solution $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective.

In the set of solutions P , the non-dominated subset of solutions P' contains those solutions that are not dominated by any member of P . The non-dominated subset of all possible solutions is called Pareto-optimal set. The goal of multiobjective optimization is to find solutions of the Pareto-optimal set.

In the past, many variants of multiobjective genetic algorithms were proposed, for example, Vector Evaluated Genetic Algorithm (VEGA), Strength Pareto Evolutionary Algorithm (SPEA), and non-dominated sorting genetic algorithm (NSGA-II) [8]. A modification of NSGA-II algorithm called the multimodal NSGA-II was previously successfully used to solve the feature selection problem – identifying a minimal subset of genes for cancer classification [9]. Multimodal algorithms are utilized in the case that many different but equally good solutions exist and it is important to find many of them.

4 Method

The proposed method can be used to either predict the traffic flow or estimate missing values for a broken sensor. In the first phase, the SVR model is trained

using historical data (train set) in the supervised learning scenario [10]. Trained SVR model then describes mathematical dependencies among the values of the sensor for which predictions are desired and other sensors in the area. Other historical data, unseen during the learning phase (test set), are used to validate the resulting model. The multiobjective multimodal NSGA-II algorithm is employed to find the proper subset of input sensors for the SVR model.

Traffic data are usually available as a set of time series s_1, \dots, s_n ; one time series for each variable measured by a traffic sensor. In order to train the SVR model, it is necessary to convert these data into training samples (Fig. 1). By means of a sliding window, the current value ($s_i^{(0)}$) and a few (h) previous values ($s_i^{(-1)}, \dots, s_i^{(-h)}$) from each series are taken into a training sample. In the case of estimating the current value of a broken sensor (Fig. 1, left), the current value $f^{(0)}$ is included into the training sample as a dependent variable. In the case of traffic forecasting in the place of sensor, the future value $f^{(+l)}$ is included into the training sample (Fig. 1, right), where l represents the prediction horizon.

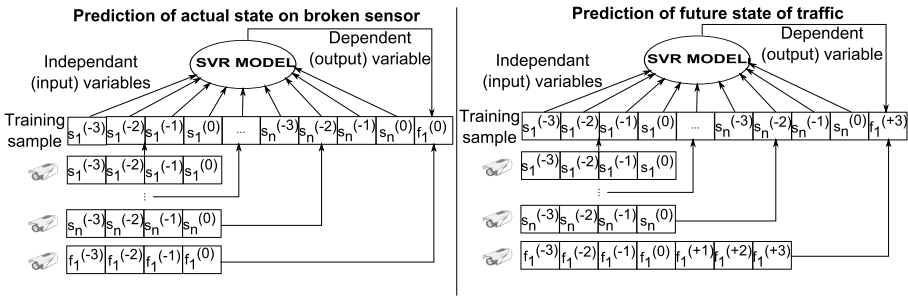


Fig. 1. Composition of training samples for SVR: prediction of a current value (left) and prediction of a future value (right) of a sensor producing f

We employed the multiobjective multimodal NSGA-II operating over binary strings. Each gene represents one input sensor, where 1 denotes including and 0 excluding of a particular sensor from the input vector fed to SVR (Fig. 2).

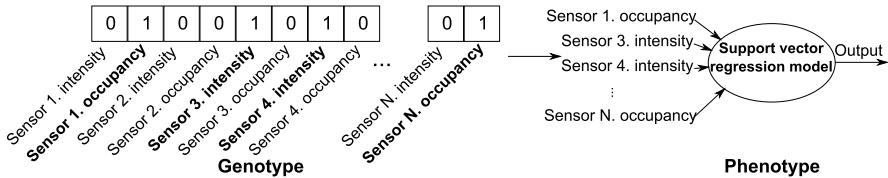


Fig. 2. Chromosome encoding and a corresponding phenotype (SVR model)

Three objectives are considered (all to be minimized) – the number of sensors used as inputs for SVR, the rate of missing samples for prediction and the

prediction error. The rate of missing samples is portion of time for which the concrete model can't be used because of missing data. All objectives are evaluated using the test set. Two well-known metrics can be used: root mean squared error (RMSE) and relative squared error (RSE) defined as

$$RMSE = \sqrt{\frac{\sum_{i=1}^d (y_i - y'_i)^2}{d}} \quad RSE = \frac{\sum_{i=1}^d (y_i - y'_i)^2}{\sum_{i=1}^d (y_i - \bar{y})^2}, \quad (2)$$

where d represents the number of regression samples, y^i is the desired value for i th regression sample and y'_i is predicted by current model. The value \bar{y} denotes the mean value predicted by a naive regression model. In the further evaluation of our method we will use RMSE as the error metric.

5 Experimental Results

5.1 Data Sets

We used publicly available data from traffic sensors in Seattle [11]. Sensors are placed on 23 intersections in the city and measure the traffic flow, occupancy and average speed. The rough data are aggregated in data tables into 1 minute intervals for a period starting on May 1st and ending on October 31st 2011. Among other information in the data tables, each row contains the traffic flow, occupancy and average speed for one sensor, and a flag indicating correctness of the measured data. All our experiments are performed using the data coming from one subarea of Seattle. Incorrect records were removed from the data tables and the remaining data were aggregated into 5 minute intervals.

5.2 SVR Parameters Setting

Although the optimization of SVR metaparameters is not the primary objective of this work, we tried to identify the most suitable setting of basic parameters of SVR which employs radial basis kernels (RBF). Figure 3 shows RSE for various settings of the regularization parameter ($C = \{2^{-5}, 2^{-4}, \dots, 2^{14}, 2^{15}\}$) and kernel parameter ($\gamma = \{2^{-15}, 2^{-14}, \dots, 2^2, 2^3\}$). In the following experiments we will utilize $C = 2^3$ and $\gamma = 2^{-12}$ because a clear minimum of RSE can be seen for them in Fig. 3.

5.3 Evaluation of NSGA-II

The proposed method was evaluated on places 6, 11, 19, 22, and 23 of the area [11]. For each sensor located on these places, four SVR models were created. The first two SVR models are trained to perform a short-term prediction in the horizon of 15 minutes. One of them uses only the actual values measured on the neighbor detectors in the area and the second one uses the actual values and the values measured on these sensors in previous 15 minutes. The other two

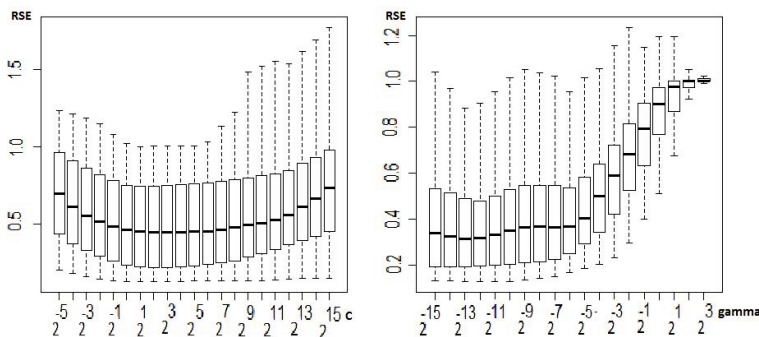


Fig. 3. The effect of setting of the regularization parameter C and kernel parameter γ on the quality of SVR prediction

SVR models are trained to estimate the actual value on the sensor in the case of a sensor error. And again, one of them uses only the actual values measured on the neighbor detectors in the area and the second one uses the actual values and the values measured on these sensors in previous 15 minutes.

The parameters of the NSGA-II genetic algorithm are as follows. The probability of uniform crossover is 70% and the probability of mutation is 5%. Each NSGA-II run, which operates with a 40 member population and 100 generations (4000 fitness evaluation), is repeated 20 times. The prediction error is given as the RMSE. The evolution utilizes approximately 50% of the available data to train SVR model, the remaining data are used to validate the evolved SVR models in the following figures and tables. Experiments were performed on an Anselm supercomputer whose nodes are equipped with two Intel Sandy Bridge E5-2665 chips. These chips contain 8-core processors working at 2.4 GHz. One run takes approximately 10 hours of one processor core. Our software was implemented in the scripting language of the system R for statistical computing [12]. We used publicly available R package e1071 for training of SVR models.

Figure 4 shows the resulting Pareto fronts from a typical NSGA-II run. Numerous non-dominated compromises between RMSE and the number of input sensors (left) and RMSE and the ratio of missing samples (right) are shown. The results were obtained for the future traffic forecasting scenario with the prediction horizon of 15 minutes for sensor number 3 measuring the traffic flow on place 19. The predicted values and correct values for one example solution are shown in Fig. 5.

Another experiment shows that the proposed method, in contrast with a common approach reported in the literature, can provide reasonable results even if many samples are unavailable. The best results obtained from 20 independent runs of NSGA-II are presented as box plots in Fig. 6. Resulting RMSE values are shown for the traffic flow and occupancy ($l = 3, h = 3$) when less than 10%, 30%, 50%, and 70% samples are unavailable. The results are given for sensor 3 on place 11. It is important to note that, for example, a value of 70% means that

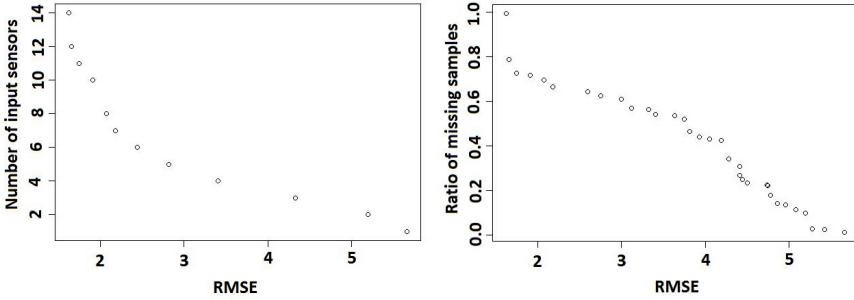


Fig. 4. Non-dominated compromises obtained for the future traffic forecasting scenario with the prediction horizon of 15 minutes for sensor number 3 measuring the traffic flow on place 19

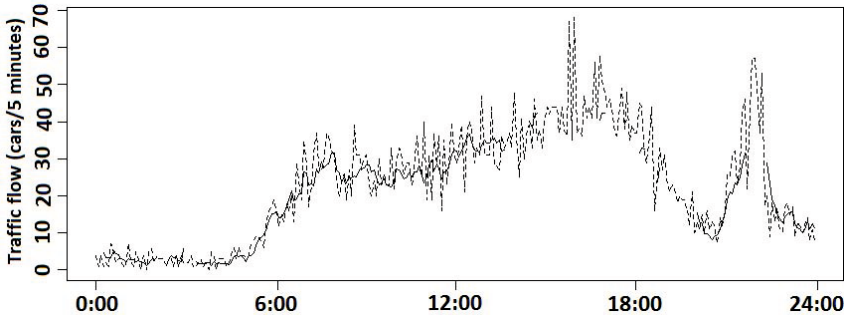


Fig. 5. Predicted values (dashed line) and correct values (normal line) of traffic flow for sensor 3 on place 19 on July 1st, 2011

for a given SVR model the samples from the test set incorrect in $24 \cdot 0.7 = 16.8$ hours of a day, i.e. the SVR model will not work for most of the time.

In order to provide results for some other sensors, Fig. 7 summarizes the best RMSE values obtained for places 6, 11, 19, 22, 23. For each place, 3 sensors exhibiting the biggest mean traffic flow and occupancy were chosen. It can be seen that RMSE increases when more samples are available in the test set. The short term traffic prediction scenario with horizon of 15 minutes and 15 minute history is considered in the figure.

And finally, Fig. 8 summarizes the mean RMSE over all sensors on all prediction places in all considered scenarios. The columns are:

- *actual* – the prediction of the actual values on a broken sensor using the actual values on sensors from other places ($l = 0, h = 0$)
- *actual (15 min.)* – see actual, but in addition, some historical data are used ($l = 0, h = 3$)

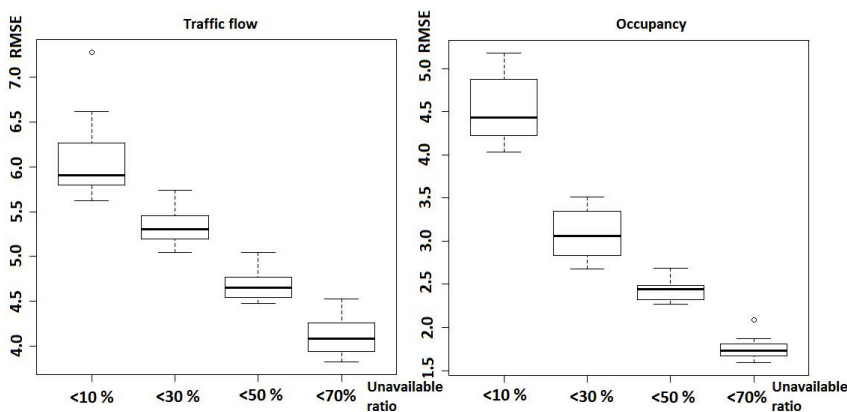


Fig. 6. Prediction error (RMSE) when less than 10%, 30%, 50%, and 70% samples are unavailable from sensor 3 on place 11

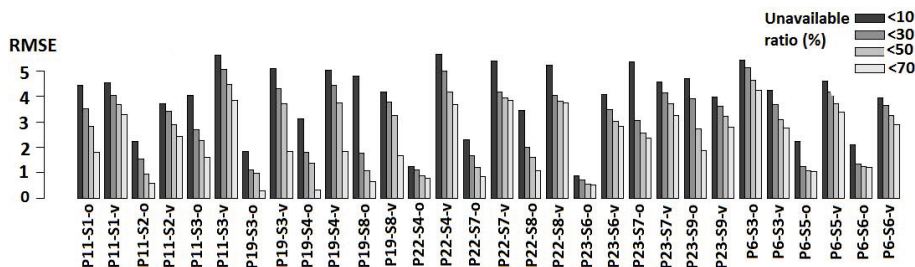


Fig. 7. Prediction error (RMSE) at 5 places (each with 3 sensors) for different amount of unavailable samples (v – traffic flow, o – occupancy)

- 15 min. future – the traffic prediction in near future with a 15 minute prediction horizon ($l = 3, h = 0$). As the input for SVR model the actual values on other sensors were used.
- 15 min. future, 15 min. history – see the previous one, but the historical data are used ($l = 3, h = 3$).

5.4 Comparison with a Single Objective GA

In order to justify the multiobjective approach, we consider a single criterion optimization scenario, in which RMSE is used as the only fitness function. The single-objective GA works with 40 individuals in the population, the probability of crossover is 70%, the probability of mutation is 5%, and 2-individual tournament selection (with elitism) is chosen. Table 1 compares NSGA-II with the single objective GA for several places and sensors (the best values from 20 independent runs are reported). It can be seen that the single objective GA tends to provide solutions with very small RMSE values; however, it opportunistically

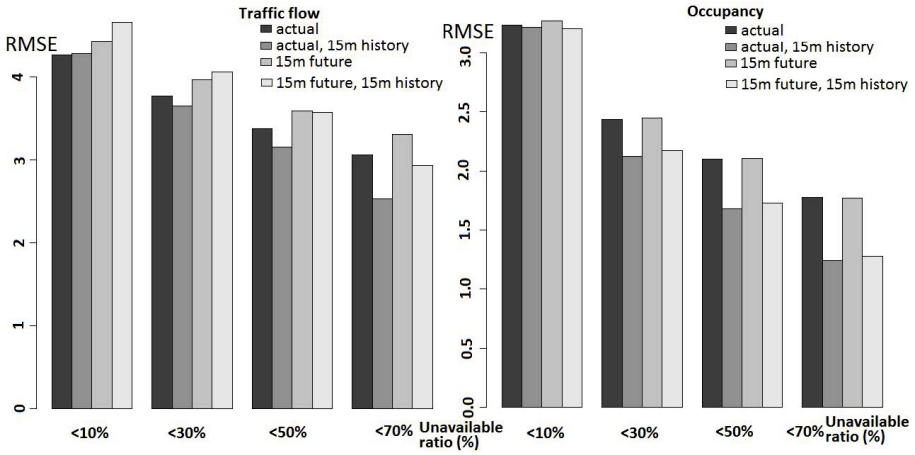


Fig. 8. Mean RMSE over all sensors on all prediction places in all considered scenarios

Table 1. The best RMSE on selected sensors and places for NSGA-II (less than 10%, 30%, 50% and 70% samples unavailable) and a single objective GA

Location			Multiobjective approach				Best single objective GA result	
Place	Sensor	Variable	RMSE for Unavailable ratio:				RMSE	Unavailable ratio
			< 10%	< 30%	< 50%	< 70%		
Current values on sensor.								
11	3	traffic flow	5.27	4.63	4.16	4.01	2.66	96.9
11	3	occupancy	3.81	3.5	3.31	3.31	0.31	99.4
22	4	traffic flow	5.33	4.86	4.31	4.2	1.48	99.4
Prediction horizon 15 min.								
11	3	traffic flow	5.5	4.9	4.37	4.23	2.96	97.2
11	3	occupancy	4.02	3.57	3.41	3.35	0.33	99.4
22	4	traffic flow	5.51	4.89	4.56	4.35	1.84	99
Current values on sensor, 15 min. history.								
11	3	traffic flow	5.2	4.58	3.91	3.34	1.15	99.4
11	3	occupancy	4.04	2.72	2.18	1.5	0.19	99.4
22	4	traffic flow	5.62	4.71	4.09	3.37	1.04	99.4
Prediction horizon 15 min., 15 min. history								
11	3	traffic flow	5.62	5.05	4.48	3.82	1.17	99.4
11	3	occupancy	4.03	2.68	2.27	1.59	0.24	99.4
22	4	traffic flow	5.64	4.98	4.17	3.66	1.15	99.4

exploits the test data containing over 85% missing values (in many cases, over 99%, see the Unavailable ratio column). Such a SVR model will thus be useless in practice, because it will not provide any prediction most of the time. Therefore, the single optimization scenario fails in this task.

6 Conclusions

In this paper, we proposed a new method for multiobjective selection of input sensors for prediction of the traffic flow. The method is based on SVR and multimodal and multiobjective NSGA-II algorithm. Contrasted to a single objective optimization scenario, in which only the prediction error has to be minimized, the multiobjective approach allowed us to find a good trade-off between the prediction error and the number of sensors in real-world situations when many traffic data measurements are not available. One can observe that adding the historical data reduces the prediction error of the occupancy prediction.

Acknowledgments. This work was supported by the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070, Brno University of Technology under number FIT-S-14-2297, and Technology Agency of the Czech Republic (TACR) project TA02030915.

References

1. Treiber, M., Kesting, A., Thiemann, C.: *Traffic Flow Dynamics: Data, Models and Simulation*. Springer (2012)
2. Dia, H.: An object-oriented neural network approach to short-term traffic forecasting. *European Journal of Operational Research* 131(2), 253–261 (2001)
3. Basak, D., Pal, S., Patranabis, D.C.: Support vector regression. *Neural Information Processing-Letters and Reviews* 11(10), 203–224 (2007)
4. Castro-Neto, M., Jeong, Y.-S., Jeong, M.-K., Han, L.D.: Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications* 36(3, pt. 2), 6164–6173 (2009)
5. Hong, W.-C.: Traffic flow forecasting by seasonal svr with chaotic simulated annealing algorithm. *Neurocomputing* 74(12-13), 2096–2107 (2011)
6. Li, M.-W., Hong, W.-C., Kang, H.-G.: Urban traffic flow forecasting using gausssvr with cat mapping, cloud model and pso hybrid algorithm. *Neurocomputing* 99(1), 230–240 (2013)
7. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley Interscience Series in Systems and Optimization. Wiley (2001)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
9. Deb, K., Reddy, A.R.: Reliable classification of two-class cancer data using evolutionary algorithms. *Biosystems* 72(1-2), 111–129 (2003)
10. Marsland, S.: *Machine Learning: An Algorithmic Perspective*. CRC (2009)
11. University of Washington Transportation Research Center, Research Data Exchange Website, Seattle Data Environment, datasets: Arterial Travel Times, www.its-rde.net (retrieved May 2013)
12. R Core Team: *A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria (2013)

Evolving DPA-Resistant Boolean Functions

Stjepan Picek^{1,2}, Lejla Batina¹, and Domagoj Jakobovic²

¹ Radboud University Nijmegen, Institute for Computing and Information Sciences
Postbus 9010, 6500 GL Nijmegen, The Netherlands

² Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb, Croatia

Abstract. Boolean functions are important primitives in cryptography. Accordingly, there exist numerous works on the methods of constructions of Boolean functions. However, the property specifying the resistance of Boolean functions against Differential Power Analysis (DPA) attacks was until now scarcely investigated and only for S-boxes. Here, we evolve Boolean functions that have higher resistance to DPA attacks than others published before by using two well-known evolutionary computation methods where genetic programming shows best performance.

Keywords: Genetic Algorithms, Genetic Programming, Boolean Functions, Cryptographic Properties, Transparency Order.

1 Introduction

In the area of private-key cryptography, one common division is to block ciphers and stream ciphers [7]. In both areas the nonlinear elements often represent the key part of the algorithm. The usual nonlinear elements are Boolean functions and S-boxes (vectorial Boolean function, i.e. a generalization of a Boolean function). In accordance with their importance, over the years there have been numerous works on constructing those nonlinear elements. Methods used in those works can be divided into algebraic constructions [5], random search [3] and heuristic methods [11] where each method has its drawbacks and benefits. Main advantages of heuristic methods are in a relatively easy addition of cryptographic properties to the evaluation functions and in results comparable with algebraic constructions. From this point on, we will concentrate only on Boolean functions and conduct the research accordingly. Here we note that if a Boolean function has n inputs, then there are 2^{2^n} Boolean functions possible. For n larger than 4 it is impractical to do an exhaustive search. Therefore, generating Boolean functions with desirable cryptographic properties is a hard problem.

1.1 Related Work

Here we mention only a small number of papers where the goal was to find optimal Boolean functions considering certain cryptographic properties.

Simulated annealing is used by McLaughlin and Clark to evolve Boolean functions that have several cryptographic properties with optimal values [10].

Aguirre et al. use evolutionary multiobjective approach to evolve Boolean functions with high nonlinearity [1]. Picek et al. use genetic programming and genetic algorithms to find Boolean functions that have good cryptographic properties [13]. Genetic algorithms are also used by Picek et al. to evolve S-boxes with good DPA (Differential Power Analysis) resistance for the AES case [12].

1.2 Our Contribution

To our best knowledge we are the first to consider a property that concerns the resistance of a Boolean function to side-channel attacks - *transparency order* [14]. Up to now, transparency order has been only vaguely investigated in the case of S-boxes but never for Boolean functions [9]. Furthermore, we give a comparison between some of the currently designed Boolean functions by means of evolutionary computation techniques and a Boolean function created by an algebraic method, used in an actual cryptographic algorithm. In our research we use genetic programming (GP) and genetic algorithms (GAs) to evolve Boolean functions with good transparency order values. Since there are no prior attempts to evaluate the resistance of a Boolean function to side-channel attacks, that makes a fair comparison more difficult.

In Section 2 we give some preliminary information on side-channel attacks and cryptographic properties of Boolean functions. Evolution of Boolean functions with GAs and GP when considering the transparency order is described in Section 3. In Section 4 we present experimental results and discussion about them. Finally, conclusion and future research is given in Section 5.

2 Preliminaries

In this section we present background details about cryptographic properties of Boolean functions and introduce basic notions of side-channel analysis.

2.1 Side-Channel Analysis

Small devices on which cryptographic algorithms are implemented, such as smart cards, have become pervasive in our lives and lots of our security and privacy-sensitive data is stored on those constrained platforms. These devices typically provide unintentional output channels, often called side channels releasing some physical leakage that relates to the operations and/or even data being processed. There are several side channels possible when considering unleashed physical information, such as power consumption or electromagnetic emanation, and we refer an interested reader to [8].

2.2 Boolean Functions

A Boolean function is in mathematics usually defined as a mapping from $\{0, 1\}^n$ to $\{0, 1\}$.

A unique representation of a Boolean function is a truth table (TT) [2]. When the total lexicographical order is assigned, Boolean function f with n inputs has a truth table with 2^n elements, where each element a fulfills $a \in \{0, 1\}$.

A Boolean function is uniquely represented by its Walsh transform which is a real-valued function defined for all $\omega \in \mathbb{F}_2^n$ as [2]

$$W_f(\omega) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus \mathbf{x} \cdot \omega}, \tag{1}$$

where $\mathbf{x} \cdot \omega$ is the scalar product of vectors \mathbf{x} and ω .

Here, \mathbb{F}_2^n is an n -dimensional vector space over Galois field with two elements [2].

The Hamming weight $HW(f)$ of a Boolean function f is the number of ones in its binary truth table [2].

Boolean function f with n inputs is balanced if its Hamming weight equals 2^{n-1} , i.e. the number of ones equals the number of zeros in the truth table [2].

The nonlinearity NL_f of a Boolean function is its minimum Hamming distance to any affine function and it can be calculated as [2]:

$$NL_f = \frac{1}{2} (2^n - \max |W_f(\omega)|). \tag{2}$$

In 2005, Prouff introduced a new cryptographic property of S-boxes: **transparency order**, which can be defined for an (n, m) -function as follows [14].

$$T_f = \max_{\beta \in \mathbb{F}_2^m} (|m - 2HW(\beta)| - \frac{1}{2^{2n} - 2^n} \sum_{\mathbf{a} \in \mathbb{F}_2^{n*}} | \sum_{\substack{\mathbf{v} \in \mathbb{F}_2^m \\ HW(\mathbf{v}) = 1}} (-1)^{\mathbf{v} \cdot \beta} W_{D_{\mathbf{a}}f}(\mathbf{0}, \mathbf{v})|). \tag{3}$$

Here, $W_{D_{\mathbf{a}}f}$ represents Walsh transform of the derivative of f with respect to a vector $\mathbf{a} \in \mathbb{F}_2^n$. This property is special since it is related with the resistance of the S-boxes to the differential power analysis attacks where higher transparency order value means lower resistance to DPA attacks [14]. Since Boolean functions are a special form of S-boxes with one output variable (therefore, $m = 1$) maximum (and the worst) possible transparency order for a Boolean function equals 1 [14].

For a Boolean function to be usable in cryptography it needs to be balanced, with high nonlinearity, high algebraic degree and high correlation immunity. For further information about these properties we refer readers to [2]. Furthermore, if one aims at better resistance against side-channel analysis then the transparency order should be as low as possible. However, Prouff showed that minimal transparency equals 0 and is achieved only in case of linear or affine functions which are not suitable for cryptography [14]. Therefore, there are two problems when creating Boolean functions with improved transparency order. The first problem is to determine an *appropriate* level of nonlinearity from a cryptographic perspective. The second problem is to find a Boolean function with at least that level of nonlinearity and with transparency order as good as possible.

3 Evolving Boolean Functions

To capture good cryptographic properties, we need to devise an appropriate fitness function to guide the evolution process. In our experiments, fitness functions incorporate properties mentioned in Sect. 2 (balancedness, nonlinearity and transparency order). There exist more properties, but since there is a trade-off between some of the them we decided to go for as simple as possible fitness function [2].

The *balancedness* property is the only one which is always strict as a part of the fitness function, since unbalanced Boolean functions are not appropriate for cryptography. The balancedness property (*BAL*) is used as a penalty, and presented in pseudo-code it calculates as

```

if ( $HW(TT) > \frac{2^n}{2}$ ) then
     $BAL = \frac{2^n - HW(TT)}{HW(TT)} \cdot X$ 
else
     $BAL = \frac{HW(TT)}{2^n - HW(TT)} \cdot X$ 
end if

```

where we experimentally find that $X = -5$ scales well for Boolean functions with $n = 8$ inputs. Balancedness is rated gradually, so that a balanced function receives the value 1, and unbalanced functions receive a negative value corresponding to the level of unbalancedness in the range r , where $r \in [-1275, -5]$. Minimum value is given when the number of ones is either 0 or 2^n .

A balanced Boolean function has an upper bound on *nonlinearity* as given in [2]:

$$NL_f = 2^{n-1} - 2^{\frac{n}{2}-1}. \quad (4)$$

This bound can be achieved only when n is even. Functions that have maximal nonlinearity are called *bent functions*, but they are not appropriate for use in cryptography since they are not balanced. Therefore, we expect to find functions with nonlinearity below 120 for $n = 8$ case.

As mentioned in Section 2, the worst possible *transparency order* equals 1. Based on the results from Mazumdar et al. who show 0.1 decrease in transparency order over the AES example as 8×8 vectorial Boolean function, we conclude that it is possible to expect the improvement in transparency order of around 1/8th of 0.1 which amounts to 0.0125 [9]. Therefore, on the basis of the data from above, we see that the nonlinearity is in range $[0, 120]$ and transparency is in range $[0, 1]$ for a balanced Boolean functions with 8 inputs.

3.1 Fitness Functions

We present here the two settings we are interested in from the experimental perspective. In the first setting the goal is to find as high as possible nonlinearity value and for that nonlinearity level the best corresponding transparency order. This is represented with the following fitness function:

$$fitness_1 = BAL + NL_f + (1 - T_f). \quad (5)$$

Given Equation (5), the optimization problem considers the *maximization* of the fitness function. Note that, once the balancedness is achieved, the main driving force of fitness is the nonlinearity, with values in excess of 100. Since the nonlinearity NL_f can only assume integer values, and transparency T_f assumes values in $[0, 1]$, it is clear that transparency is used as a secondary objective, to be able to perform selection of different solutions with the same level of nonlinearity.

Since there are three different properties to optimize, one possible approach would be the use of multiobjective optimization. We do not use that approach for several reasons; first of all, the balancedness is an absolute requirement and should not be included as an independent measure. On the other hand, we have to differentiate the unbalanced solutions in the evolution and allow them to improve, so a part of the fitness function (BAL) is devised to produce the greatest penalty. The second most important property is nonlinearity, for which we want it to be as high as possible. Since we do not know in advance what nonlinearity is achievable, the algorithm should be driven to find the maximum value, regardless of the other properties. Only when this is evolved, we need to find solutions with the transparency order as good as possible. Since the nonlinearity can only assume a fixed number of levels, the search can be adjusted to suit those values, which is investigated in the following setting.

In the second setting the goal is to find low transparency order values for a certain nonlinearity level. Here we reiterate that the lowest possible transparency order is achieved for linear and affine functions, but since they are not appropriate for the use in cryptography we do not consider them as a viable choice [2]. Because it is difficult to say what would be an appropriate nonlinearity value, we decided to set several levels as a constraint. Therefore, we look for the best transparency order with a target minimum nonlinearity level, NL_t . We set 4 levels of nonlinearity, at the values of 86, 92, 98 and 104. Fitness function for the second set of experiments is defined as:

$$fitness_2 = BAL + (1 - T_f) - pos(2 \times (NL_t - NL_f)), \quad (6)$$

where the function $pos(x)$ returns x if $x > 0$ and zero otherwise.

3.2 Algorithms, Representations and Parameters

As previously mentioned there are several ways to uniquely represent Boolean functions. For GA we decided to represent the individuals as strings of bits where values are truth tables of functions, and GP individuals as trees of Boolean primitives which are then evaluated according to the truth table they produce. We use two selection mechanisms, namely steady-state with tournament operator (SST) and generational with roulette-wheel (RW) selection. In the first one, 3 solutions are selected at random and the worst among them is replaced by the offspring of the remaining two. The offspring is mutated with a given probability (0.3).

The RW selection uses the fitness proportional roulette-wheel operator applied to the whole population to select the new generation of survivors, to which crossover

and mutation are then applied; in this scenario, the offspring replaces the parents. The same two selections are applied both to GA and GP. All the employed methods are a part of the Evolutionary Computation Framework (ECF) [6].

In the experiments we initialize population with random individuals where we do not require that the solutions are balanced. That is opposite from what is usually done [4, 11], but we expect that the evolution process would benefit from that additional diversity.

In an effort to find the differences in the performance of the algorithms, we also compare the best algorithms with the *balanced random search* algorithm. Balanced random search uses random sampling of solutions, but only considering balanced Boolean function as a solution candidate, as opposed to a purely random choice of solutions. Stopping criterion for the random search algorithm is 200 000 evaluations.

GA Variations. For GA representation, mutation is selected uniformly at random between simple mutation, where a single bit is inverted, and mixed mutation, which randomly shuffles the bits in a randomly selected subset. Additionally, we use a *balanced mutation* that preserves balancedness of the solution by changing 2 bits of the individual if it is already balanced.

For all mutation operators we experiment also with an *adaptive mutation rate*. In the beginning, the mutation is given as a fixed probability (0.3), but as the evolution starts to stagnate (i.e. no improvement in the best solution), the mutation probability raises. The probability is increased linearly with the number of generations without improvement, until it reaches a predefined maximum level (0.8) after a given maximum number of generations without improvement has passed. If a new best solution is found, the mutation rate is reset to initial value. The crossover operators used in GA are one-point and uniform crossover, performed at random for each new offspring.

GP Variations. Of the modifications in the previous section, with GP we employ only the adaptive mutation rate, in the same manner as for the GA. The function set for genetic programming in all the experiments is OR, NOT, XOR, AND, IF, and terminals correspond to 8 Boolean variables. Genetic programming has maximum tree depth of 11. For the Boolean functions we are interested only in XOR and AND operators, but it is quite easy to transform the function from one notation to the other.

Common Parameters. Parameters that are in common for every round of the experiments are the following: the size of Boolean function is 8 (the size of the truth table is 256) and the population size is 500. In the roulette-wheel selection the crossover probability is 0.5 and the ratio of the probability of survival of the best and the worst individual is scaled to 10. Mutation probability for non-adaptive variations is set to 0.3 per individual. The parameters above are the result of a combination of a small number of preliminary experiments and our experience with similar problems; no thorough parameter tuning has been performed.

Unlike in the traditional optimization case, our main goal is not to compare different approaches we implement, but rather to find the best possible solutions. In that case, we do not limit the algorithms with a fixed number of evaluations (and compare the averages), but allow the algorithms to run as long as something useful might occur. That is why the stopping criterion is set to a given number of generations without improvement, rather than a fixed maximum number, which is a reasonable criterion for researchers trying to find an adequate solution.

Further information regarding experimental setup is listed when needed.

4 Results and Discussion

When presenting the best achieved results, we also compare them with some of the existing results from the literature. For previously published Boolean functions, we use the following notation: for the Boolean function from the Burnett et al. paper we use the name Function_1 [4], from the work by McLaughlin and Clark we abbreviate Boolean function with Function_2 [10]. For the Boolean function from the work by Picek et al. we abbreviate it with Function_3 [13], and finally, for Rakaposhi Boolean function we abbreviate it with Rakaposhi [5].

Here we emphasize that Boolean function in Rakaposhi cipher is obtained through algebraic construction, more specifically the finite field inversion method. Rakaposhi stream cipher is an example of a modern, state-of-the-art stream cipher that uses Boolean function as a nonlinear element. The algorithm names presented in tables are abbreviated in the following way: after the abbreviation of the algorithm we write in the subscript the distinguishing properties of the algorithm: *SST* represents steady state tournament selection, *RW* roulette wheel selection, *balanced* means balanced mutation operator and *variable* represents variable mutation rate.

With the objective to find the best individuals, the stopping criterion is set to 50 generations without improvement for all algorithm variations and the number of independent runs is 400. In Table 1 we give the best result for each of the experiments conducted, as well as for random search algorithm and Boolean functions from related works. Here, evolutionary algorithms use fitness function as defined by Equation (5). All solutions are balanced so we did not specifically write that property.

Table 1. Best Boolean functions, $fitness_1$

Algorithm	NL_f	T_f	Algorithm	NL_f	T_f
<i>GP_{SST}</i>	116	0.962	<i>GP_{SST,variable}</i>	112	0.919
<i>GP_{RW,variable}</i>	112	0.965	<i>GA_{SST}</i>	112	0.934
<i>GA_{SST,balanced,variable}</i>	112	0.931	<i>GA_{SST,balanced}</i>	112	0.938
<i>GA_{RW,balanced}</i>	112	0.935	<i>Random search</i>	110	0.934
<i>Function_1</i>	100	0.927	<i>Function_2</i>	116	0.969
<i>Function_3</i>	116	0.976	<i>Rakaposhi</i>	112	0.946

Based on the related work, we choose nonlinearity levels of 112 and 116 as the most interesting ones. The best (lowest) transparency results for those nonlinearity levels are in bold style.

When optimizing the second fitness function (6), we concentrate on 4 pre-defined nonlinearity levels. Here we do not compare this results with literature since previous works did not investigate transparency property (and therefore those works can have better nonlinearity in general, but when looking at both of those properties related works have worse results). In this case we are interested in the lowest transparency value that can be obtained with a given minimum nonlinearity. For $fitness_2$ equation the best results are presented in Table 2.

Table 2. Best Boolean functions, $fitness_2$

Algorithm	86	92	98	104
<i>GP_{SST,variable}</i>	0.774	0.815	0.866	0.898
<i>GA_{SST,balanced,variable}</i>	0.78	0.817	0.822	0.866
<i>Random Boolean</i>	0.887	0.894	0.905	0.915

The space of possible Boolean functions is huge and therefore it is impractical to do an exhaustive search for Boolean functions with the number of inputs relevant in cryptography. However, the space of Boolean functions with “good” cryptographic properties is also large. In such a large space it is difficult to find Boolean functions with excellent cryptographic properties. This is an obvious example of the convergence of the algorithm towards the local optima. In an attempt to search beyond those local optima we employ different algorithms and modifications.

In the experiments we use two different selection methods where we expected that the roulette-wheel selection should be the best one, because preliminary results (also the results from other researchers) showed that all algorithms display very quick convergence. However, algorithms with the steady-state tournament selection consistently found the best solutions among all the algorithms.

A simple fitness function, that includes only a subset of the desired properties, has the advantage that there are no conflicts between variables, and some high quality properties inherently mean that other properties will also be good. In our previous experiments, preliminary results show that it is not trivial to combine many properties in a single fitness value because of varying magnitudes (scaling issues) of different property values.

Statistical analysis (not presented in the paper) shows that all GA variations give similar results with smaller standard deviation than in GP variations. We also conducted pairwise comparison between GP algorithm with steady-state tournament selection and all other algorithms where the results show there are no statistically significant differences. However, genetic programming with steady-state tournament produced the single best result when considering the nonlinearity value. For a nonlinearity level 112 the best result was achieved with

$GP_{SST,variable}$ algorithm. In this case we can also see that this Boolean function has better transparency order value than the Rakaposhi Boolean function.

As evident from the results, we found Boolean functions with better transparency order values but the improvements are rather small. One could ask if such small improvements make a difference. We consider this to be true, since we did not expect obtaining a big difference if the analogy with S-boxes (where 0.1 is a significant improvement) is valid. Furthermore, side-channel analysis of stream ciphers is more difficult and therefore even small improvements are relevant. Naturally, further investigation of the relevance of the transparency order property is needed in order to put these results in proper perspective. Our new Boolean functions present viable choice for future implementations since they are offering improvement in the properties while not bringing additional area or speed drawbacks.

5 Conclusions and Future Work

Finding Boolean functions with improved DPA resistance is a difficult problem, not only because of the huge search space, but also due to the lack of previous work. As far as we know, we are the first to experiment with the transparency order property for Boolean functions and additionally to use evolutionary algorithms to find suitable functions. Our experiments showed that it is possible to find Boolean functions with better transparency order than that in reference work. From cryptographic perspective, genetic programming achieve the highest nonlinearity level with an improved transparency order value. However, from the evolutionary point of view, the results show no statistically significant difference with respect to the genetic algorithm. Due to the lack of prior work on this topic, results obtained here should be also regarded as a baseline for future research.

As future research directions we plan to experiment with other algorithms like Cartesian GP or Estimation of Distribution. Initial experiments give promising results.

Acknowledgments. This work was supported in part by the Technology Foundation STW (project 12624 - SIDES), The Netherlands Organization for Scientific Research NWO (project ProFIL 628.001.007) and the ICT COST action IC1204 TRUDEVICE.

References

1. Aguirre, H., Okazaki, H., Fuwa, Y.: An evolutionary multiobjective approach to design highly non-linear boolean functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2007, pp. 749–756 (2007)
2. Braeken, A.: Cryptographic Properties of Boolean Functions and S-Boxes. PhD thesis, Katholieke Universiteit Leuven (2006)
3. Burnett, L.: Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. PhD thesis, Faculty of Information Technology, Queensland University of Technology (2005)

4. Burnett, L., Millan, W., Dawson, E., Clark, A.: Simpler methods for generating better boolean functions with good cryptographic properties. *Australasian Journal of Combinatorics* 29, 231–247 (2004)
5. Cid, C., Kiyomoto, S., Kurihara, J.: The RAKAPOSHI Stream Cipher. In: Qing, S., Mitchell, C.J., Wang, G. (eds.) *ICICS 2009*. LNCS, vol. 5927, pp. 32–46. Springer, Heidelberg (2009)
6. Jakobovic, D., et al.: Evolutionary computation framework (December 2013), <http://gp.zemris.fer.hr/ecf/>
7. Katz, J., Lindell, Y.: *Introduction to Modern Cryptography*. Chapman and Hall/CRC, Boca Raton (2008)
8. Mangard, S., Oswald, E., Popp, T.: *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Advances in Information Security. Springer-Verlag New York, Inc., Secaucus (2007)
9. Mazumdar, B., Mukhopadhyay, D., Sengupta, I.: Constrained Search for a Class of Good Bijective S-Boxes with Improved DPA Resistivity. *IEEE Transactions on Information Forensics and Security* PP(99), 1 (2013)
10. McLaughlin, J., Clark, J.A.: Evolving balanced boolean functions with optimal resistance to algebraic and fast algebraic attacks, maximal algebraic degree, and very high nonlinearity. *Cryptology ePrint Archive*, Report 2013/011 (2013), <http://eprint.iacr.org/>
11. Millan, W.L., Clark, A.J., Dawson, E.: Heuristic design of cryptographically strong balanced boolean functions. In: Nyberg, K. (ed.) *EUROCRYPT 1998*. LNCS, vol. 1403, pp. 489–499. Springer, Heidelberg (1998)
12. Picek, S., Ege, B., Batina, L., Jakobovic, D., Chmielewski, L., Golub, M.: On Using Genetic Algorithms for Intrinsic Side-channel Resistance: The Case of AES S-box. In: *Proceedings of the First Workshop on Cryptography and Security in Computing Systems, CS2 2014*, pp. 13–18. ACM, New York (2014)
13. Picek, S., Jakobovic, D., Golub, M.: Evolving Cryptographically Sound Boolean Functions. In: *GECCO (Companion)*, pp. 191–192 (2013)
14. Prouff, E.: DPA Attacks and S-Boxes. In: Gilbert, H., Handschuh, H. (eds.) *FSE 2005*. LNCS, vol. 3557, pp. 424–441. Springer, Heidelberg (2005)

Combining Evolutionary Computation and Algebraic Constructions to Find Cryptography-Relevant Boolean Functions

Stjepan Picek^{1,2}, Elena Marchiori¹, Lejla Batina¹, and Domagoj Jakobovic²

¹ Radboud University Nijmegen, Institute for Computing and Information Sciences
Postbus 9010, 6500 GL Nijmegen, The Netherlands

² Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb, Croatia

Abstract. Boolean functions play a central role in security applications because they constitute one of the basic primitives for modern cryptographic services. In the last decades, research on Boolean functions has been boosted due to the importance of security in many diverse public systems relying on such technology. A main focus is to find Boolean functions with specific properties. An open problem in this context is to find a balanced Boolean function with an 8-bit input and nonlinearity 118. Theoretically, such a function has been shown to exist, but it has not been found yet. In this work we focus on specific classes of Boolean functions, and analyze the landscape of results obtained by integrating algebraic and evolutionary computation (EC) based approaches. Results indicate that combinations of these approaches give better results although not reaching 118 nonlinearity.

Keywords: Boolean Functions, Nonlinearity, Evolutionary Computation, Bent Functions, Cryptographic Properties.

1 Introduction

Cryptography is crucial in most security applications by helping cryptographic services to achieve secure communication through unsecured channels. The main goal is to secure messages so that only the relevant parties can read them. A message (plaintext) is transformed into an incomprehensible form (ciphertext) by a process called encryption, while an encrypted message is mapped to its original form through a process called decryption. Encryption and decryption are performed using symmetric key algorithms. Boolean functions constitute one of the basic primitives for symmetric key algorithms, most notably in stream ciphers [1]. Stream ciphers, based on an input key, generate a sequence of random bits which is used as keystream that will never be used again during the run of the cipher. Boolean functions for cryptography must satisfy various possibly contrasting properties, such as being balanced, highly nonlinear, correlation immune, t -resilient. Therefore, it is typically hard if not impossible to find an optimal function [1,2]. Finding Boolean functions for cryptography is clearly a

challenging problem, because there are 2^{2^n} possible Boolean functions of n inputs (for instance, when n equals 8 which is usual size in today cryptography, this gives 2^{256} candidate solutions). There are three main approaches to generate Boolean functions for cryptography: algebraic construction, random generation and heuristic construction. Algebraic construction employs well established mathematical procedures that give very good results [3]. Random generation of Boolean functions has its advantages, most prominent one being easy and fast construction, but the resulting Boolean functions usually have suboptimal properties for cryptography [4]. Heuristic methods provide a relatively easy and efficient way of producing large number of Boolean functions with very good cryptographic properties [2]. In particular, Evolutionary computation (EC) has been successfully applied to evolve Boolean functions for cryptography as listed below. So far, researchers on Boolean function generation for cryptography have focused on the optimization of some cryptographic properties. In this work we focus on specific classes of Boolean functions, and analyze the “landscape” of results generated using and integrating algebraic and EC based approaches, in an effort to gain deeper insights on their individual and joint performance.

Specifically we focus on an open problem in cryptography: find a balanced 8-bit Boolean function with nonlinearity 118. Although theoretically it has been proved that such a function exists, no one has succeeded in finding it. Our goal is to investigate properties of EC methods based on the integration of the above mentioned approaches, in particular how difficult is to find Boolean functions with certain set of properties (for instance bent Boolean functions of a specific size) and what is the influence of a specific initial population (consisting of previously evolved bent Boolean functions).

1.1 Our Contributions

Besides evolving balanced function with 118 nonlinearity, our experiments investigate hardness of evolving bent Boolean functions and functions that can be used in algebraic constructions. As far as the authors know, we are the first to investigate the evolution of 7-bit functions and to give insights about the hardness of that problem and distribution of resulting Boolean functions. Our hybrid methods prove to be much more successful in evolving highly nonlinear balanced functions than simple evolution methods.

The remainder of this paper is organized as follows: after a short overview of related works, in Section 2 we describe the problem and relevant properties of Boolean functions. In Section 3 the considered methods are described, and in Section 4 experimental setup and results are given. Finally, Section 5 concludes with some suggestions for future work.

1.2 Related Work

We distinguish two main categories of methods for generating Boolean functions with properties of interest: those dealing with algebraic constructions based on

mathematical results and those dealing with heuristic methods. For each of these categories we give only a short overview of work related to our investigation.

Algebraic Constructions. Sarkar and Maitra propose new construction methods which were used to obtain functions that were not known earlier [5]. In the same year, those authors also present a theorem that states stricter upper bound on nonlinearity of resilient Boolean functions [6]. Zheng and Zhang improve upper bound of the nonlinearity of high order correlation immune functions [7]. Pasalic et al [8] construct several functions that were not known before that have upper bound on nonlinearity. Those functions were stipulated to exist due to the paper of Sarkar and Maitra [6].

Evolutionary Computation. Aguirre et al. use evolutionary multiobjective approach to evolve Boolean functions that have high nonlinearity [9]. Clark et al. use simulated annealing to find Boolean functions that satisfy several properties desired for cryptographic usage [10]. Burnett et al. create two heuristic methods to evolve Boolean functions for usage in cryptography. Picek et al. use genetic programming and genetic algorithms to find Boolean functions with cryptographic properties [11].

2 Preliminaries

In this section we describe relevant cryptographic properties of Boolean functions and theoretical results that inspired our investigation. In the sequel $\mathbf{a} \cdot \mathbf{b}$ denotes the inner product of vectors \mathbf{a} and \mathbf{b} defined as $\bigoplus_{i=1}^n a_i b_i$, where “ \oplus ” denotes addition modulo 2.

2.1 Representations and Properties

A Boolean function f on \mathbb{F}_2^n can be uniquely represented by a truth table (TT), which is a vector $(f(\mathbf{0}), \dots, f(\mathbf{1}))$ that contains the function values of f , ordered lexicographically [1].

Walsh transform is a second type of unique representation of a Boolean function. It measures the similarity between $f(\mathbf{x})$ and the linear function $\mathbf{a} \cdot \mathbf{x}$ [1]. Walsh transform of a Boolean functions f equals

$$W_f(\mathbf{a}) = \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{f(\mathbf{x}) \oplus \mathbf{a} \cdot \mathbf{x}}. \quad (1)$$

Third unique representation of an Boolean function f on \mathbb{F}_2^n is by means of a polynomial in $\mathbb{F}_2[x_0, \dots, x_{n-1}] / (x_0^2 - x_0, \dots, x_{n-1}^2 - x_{n-1})$. This form is called algebraic normal form (ANF) [1]. Algebraic normal form is the multivariate polynomial P defined in [1] as:

$$P(\mathbf{x}) = \bigoplus_{\mathbf{a} \in \mathbb{F}_2^n} h(\mathbf{a}) \cdot \mathbf{x}^{\mathbf{a}}. \quad (2)$$

Next, we present some of the properties of Boolean functions that play an important role in security algorithms: nonlinearity, bent, correlation immunity, balancedness, t -resiliency and algebraic degree.

The **nonlinearity** NL_f of a Boolean function f can be expressed in terms of the Walsh coefficients as [1]

$$NL_f = 2^{n-1} - \frac{1}{2} \max_{\mathbf{a} \in \mathbb{F}_2^n} |W_f(\mathbf{a})|. \tag{3}$$

Boolean function f is **bent** if it has maximum nonlinearity equal to [1]

$$NL_f = 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{4}$$

Boolean function f is **correlation immune** of order t - $CI(t)$ if the output of the function is statistically independent of the combination of any t of its inputs [1]. Boolean function f is **t -resilient** if it is balanced and with correlation immunity of degree t [1].

A Boolean function is **balanced** (BAL) if its Hamming weight is equal to 2^{n-1} [1].

From this point on, when we talk about Boolean functions we consider them t -resilient (i.e. balanced) if not specified otherwise.

Algebraic degree $deg(f)$ of a Boolean function f , is defined as the number of variables in the largest product term of the functions' algebraic normal form (ANF) having a non-zero coefficient [2].

2.2 Theoretical Background

Sarkar and Maitra showed that if a t -resilient Boolean function f has an even number of inputs n and $t + 1 \leq \frac{n}{2} - 1$ then its nonlinearity NL_f is bounded as follows [6]:

$$NL_f \leq 2^{n-1} - 2^{\frac{n}{2}-1} - 2^{t+1}. \tag{5}$$

From the formula it follows that the maximum nonlinearity for $n = 8$ and $t = 0$ equals 118.

In that same paper the authors gave a second theorem that inspired one of our experiments.

Suppose f is a Boolean function with 8 inputs, resilience 0 and nonlinearity 118. Then degree deg of f must be 7 and it is possible to write

$$f = (1 \oplus X_8)f_1 \oplus X_8f_2 \tag{6}$$

where f_1 and f_2 are Boolean functions with 7 inputs, nonlinearity 55 and algebraic degree 7 [6].

An interesting implication of this result is that if it is not possible to construct an 8-bit Boolean function with nonlinearity 118, degree 7 and resilience 0 by concatenating two 7 inputs Boolean functions with nonlinearity 55 and degree 7 then the maximum nonlinearity of 8 inputs Boolean function is 116 [6].

3 Approach and Methods

Recall that we focus on the open problem of finding an 8-bit Boolean function with nonlinearity 118. We adopt a methodology consisting of two main phases.

In the first phase, we consider EC methods and their hybridization with algebraic constructions.

- **Simple evolution.** To analyze the effectiveness of EC methods we apply them directly to evolve 8-bit Boolean function with nonlinearity 118.
- **Bent functions.** Because bent functions have high maximum nonlinearity (see Equation (4)), in this setting we search for 8-bit bent functions. We do this in two ways: (a) using EC methods to directly evolve 8-bit bent functions, and (b) using EC methods to evolve 6-bit bent Boolean functions and use them to construct 8-bit bent functions by means of algebraic technique [12]. Furthermore, we use the resulting bent functions to seed the initial population of a EC algorithm for finding an 8-bit Boolean function with nonlinearity 118. This allows us to gain insight in the influence of the bent functions as initial population.
- **Algebraic concatenation.** In this setting we evolve 7-bit functions with nonlinearity 55 and degree 7. Then we combine and concatenate those functions using Equation (6). Note that 7-bit functions are not balanced.

In the second phase, we select the best performing algorithm and investigate algorithmic hybridizations based on combinations of the algorithms and bent Boolean functions.

The above methodology allows us to address the following questions related to the effectiveness of EC for evolving Boolean functions.

- How difficult is to find bent Boolean functions of various sizes?
- What is the influence of the initial population in the evolution of balanced Boolean functions?
- How hard is to find Boolean functions with certain set of properties that are useful in further search process?

We describe below the techniques employed in our investigation.

3.1 Algorithms, Representations, and Fitness Functions

We consider three EC methods, namely genetic algorithms (GAs), genetic programming (GP) and genetic annealing (GAn). All the employed methods are a part of the Evolutionary Computation Framework (ECF) [13].

Genetic Algorithm. We use a simple genetic algorithm with 3-tournament selection [14], where in each iteration 3 solutions are selected randomly, and the worst of those is replaced with the crossover offspring of the remaining two. Mutation is performed on the offspring, using randomly simple mutation, where a single bit is inverted, and mixed mutation, which randomly shuffles the bits in a randomly selected subset. The crossover operators are one-point and uniform crossover, performed at random for each new offspring.

Genetic Programming. The function set for genetic programming used in all experiments consists of Boolean functions OR, XOR, AND (taking two arguments), NOT (one argument) and IF (which takes three arguments and returns

the second argument if the first one evaluates to “true”, and the third one otherwise). The terminals correspond to n Boolean variables. Genetic programming has maximum tree depth of 11. Boolean functions can be expressed only in XOR and AND operators, but it is quite easy to transform from one representation to another. Genetic programming uses the same selection as the GA, with simple subtree crossover and subtree mutation.

Genetic Annealing. Genetic annealing is an evolutionary extension of the Simulated Annealing algorithm (SA) [15]. The SA operates on a single potential solution, which is locally changed in each iteration and its new fitness value is recorded. The new solution is accepted if it is an improvement, but a worse solution can also be accepted provided a certain level of *global energy bank*, which depletes with worse and increases with better solutions. The GAn is a simple extension in which the whole process is performed on a population of individuals.

Local Search Algorithm. For local search algorithm we chose strong hill climbing (HC) algorithm where all possible combinations of changes of 2 complementary bits are investigated. This process is repeated until there is no more improvement for every individual in population. Hill climbing algorithm uses truth table representation which means when using GP, individuals need to be transformed from tree representation to truth table representation. Additionally, we change 2 bits since initial solutions are balanced and we need to preserve it by changing one bit from 0 to 1, and other bit from 1 to 0.

Representations. As mentioned in Section 2, there are several ways to uniquely represent Boolean functions. For genetic algorithms and genetic annealing we decided to represent the individuals as strings of bits where values are truth tables of functions; for genetic programming, individuals are trees of Boolean primitives which are then evaluated according to the truth table they produce.

Fitness Functions. To evolve bent functions of different input sizes we consider the *maximization* of the following simple fitness function.

$$fitness = NL_f. \quad (7)$$

When searching for 7-bit functions with nonlinearity 55 and degree 7 we consider the *maximization* of the fitness function:

$$fitness = NL_f + deg. \quad (8)$$

When searching for 8-bit balanced function with as good as possible nonlinearity we try to *maximize* the following objective function:

$$fitness = BAL + NL_f. \quad (9)$$

When calculating balancedness property, we assign it to a value 1 when the function is balanced, otherwise we assign it the difference up to the balancedness multiplied with a constant, which was set to 5 after a short parameter tuning phase.

4 Experimental Setup and Results

Parameters that are common for every algorithm are the following: the size of Boolean function is 8 (the size of the truth table is 256) and the population size is 500. Mutation probability is set to 0.3 per individual. The number of independent runs for each experiment is 2000; we are aware that 30-50 is sufficient for statistical purposes, but our main concern is not a statistical comparison of algorithms, but finding the best possible solutions. Furthermore, we use the rate of successful runs (finding the optimum), rather than the average fitness, as a metric for algorithm comparison.

4.1 Results

Due to the lack of space we give only a short overview of results from phase 1. We mention that some of those results deserve additional experiments on its own; EC methods are not often used to generate bent functions and there is a clear lack of literature of how difficult it is or what methods perform the best. Furthermore, we are not aware that anyone before tried to evolve 7-bit functions with nonlinearity 55 and degree 7. In Table 1 we give results for phase 1 experiments. Column SUCCESS represents the percentage of runs that the algorithm reaches maximum value and column NL_f represents maximum nonlinearity reached by the algorithm. Naturally, we tested a random search method on all these problems but got no SUCCESS results. Based on this results we selected GP as the algorithm of choice for phase 2 as we can see that GP has best results when looking for functions with 8 inputs.

From the results for Equation (9) we see that evolving functions with nonlinearity 55 and degree 7 is easy. However, our experiments show that the level of unbalancedness differ significantly as shown in Figure 1. Therefore, it is not

Table 1. Phase 1 experiments

Algorithm, fitness, size	NL_f	Min	Max	Mean	Stdev	SUCCESS(%)
GA, (8), 6-bit	28	26	28	26.141	0.512	7.1
GAn, (8), 6-bit	28	28	28	28	0	100
GP, (8), 6-bit	28	26	28	27.382	0.924	69.1
GA, (8), 8-bit	116	112	116	113.157	0.975	0
GAn, (8), 8-bit	114	112	114	113.9	0.1	0
GP, (8), 8-bit	120	112	120	114.5	2.318	13.8
GA, (9), 7-bit	55	60	62	61.97	0.22	98
GAn, (9), 7-bit	55	60	62	61.63	0.77	81.6
GP, (9), 7-bit	55	60	62	60.06	0.33	3
GA, (10), 8-bit	114	113	115	113.524	0.879	52.4
GAn, (10), 8-bit	114	113	115	113.03	0.25	3.2
GP, (10), 8-bit	116	109	117	112.23	1.01	0.5
HC, (10), 8-bit	112	103	113	108.02	1.43	0.5

easy to find two unbalanced 7-bit functions that produce a balanced 8-bit function. For example, if we find a function with desired properties that has 69 zeros and 59 ones, then the other function needs to have 59 zeros and 69 ones (this function should not be affine equivalent to the first one since then it will have nonlinearity of 110) to produce a balanced 8-bit function. Since there were no prior experiments on this topic, these levels of unbalancedness present significant guideline for future work. With the concatenation method we obtained a balanced function with maximum nonlinearity of 110.

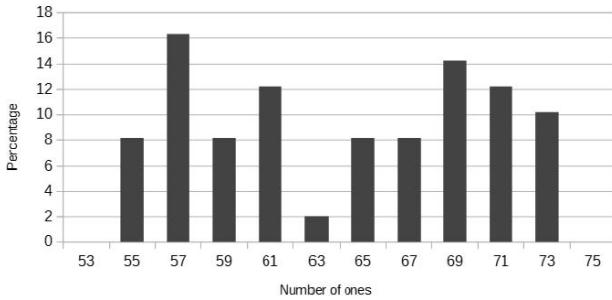


Fig. 1. Distribution of 7-bit unbalanced Boolean functions

In the second phase we use the best algorithm (GP) from the first phase when looking on the ways to further improve solutions. In this phase we concentrate only on Eq. (10) and function size 8. As the first experiment (I) in the second phase we run GP algorithm that has initial population made from bent functions from phase 1. We do not distinguish here between bent functions with 8-bit size that are directly evolved from Boolean functions with 6-bit size that are evolved and then algebraically expanded to 8-bit size. The second experiment (II) uses results from the first experiment and performs the hill climbing algorithm. In the last, third experiment (III) we run GA on the results from the first experiment. In Table 2 we give results for hybridizations of algorithms where we can see that these approaches improve algorithm behavior and percentage of the obtained maximum values. The approach that combines solutions from GP with bent functions as initial population and GA gives the best result.

EC methods we investigated did not succeed in finding a 8-bit balanced Boolean function with nonlinearity 118. So this problem remains open. Results

Table 2. Phase 2 - algorithm combinations

Experiment	NL_f	Min	Max	Mean	Stdev	SUCCESS(%)
I (GP+seed)	116	113	117	113.55	1.379	13.8
II (GP+seed+HC)	116	113	117	113.8	0.91	19.6
III (GP+seed+GA)	116	113	117	115.32	0.43	67.1

indicate that GP performs best. Not only did it find solutions with highest nonlinearity but it also resulted in final population with greatest diversity. Bent functions are not suitable for the use in cryptography and therefore not usually defined as the goal of evolutionary search. Previous works that evolved bent Boolean functions include [4, 16]. Our results show that evolving 6-bit bent Boolean functions is relatively easy because each of the algorithms obtained numerous global optima. Here, GA behaves interestingly since it always reached the optimum although many of the obtained solutions are the same. When considering 8-bit bent functions, only GP is capable to find them.

When looking at 7-bit unbalanced functions and Eq. (9), we see that GA performs best. However, it is not possible to concatenate every of those solutions since the resulting functions must be balanced. Prior to this research, as far as the authors know, there were no reports on the level of unbalancedness these functions can reach or on their distribution. From this perspective, this research provides new insights to guide future work.

When setting bent Boolean functions as the initial population for GP, we see that we are still not able to reach nonlinearity 118 but the percentage of the population with the best currently known nonlinearity (116) significantly increases. With a larger population a larger number of unique solutions is obtained. We recommend this procedure for researchers interested to use functions that were not previously known with nonlinearity 116.

The rationale behind HC algorithm was not only to show the feasibility of improving solutions previously obtained with GP, but also to confirm that there is no 118 nonlinearity in the “proximity” of solutions with nonlinearity 116. We see that average value of the final population slightly improved. When repeating the same experiment, but using GA instead of HC we see that this setting was also unable to find 118 nonlinearity function. However, the provided summary statistics show that this setting performs much better. Therefore, it seems beneficial to combine multiple EC algorithms where the initial population for one algorithm is the final population of other EC algorithm.

5 Conclusions and Future Work

In this research we had two goals; one was to find balanced Boolean function with 118 nonlinearity and the second one was to investigate the strength of EC methods hybridized with algebraic techniques for this task. Although we did not find an 8-bit function with nonlinearity 118 our results provide new insights in the area of evolving Boolean functions. As future work we are interested in fitness functions where additional variable is Hamming distance between the solutions.

Acknowledgments. This work was supported in part by the Technology Foundation STW (project 12624 - SIDES), The Netherlands Organization for Scientific Research NWO (project ProFIL 628.001.007) and the ICT COST action IC1204 TRUDEVICE.

References

1. Braeken, A.: Cryptographic Properties of Boolean Functions and S-Boxes. PhD thesis, Katholieke Universiteit Leuven (2006)
2. Burnett, L.D.: Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. PhD thesis, Queensland University of Technology (2005)
3. Cid, C., Kiyomoto, S., Kurihara, J.: The RAKAPOSHI Stream Cipher. In: Qing, S., Mitchell, C.J., Wang, G. (eds.) ICICS 2009. LNCS, vol. 5927, pp. 32–46. Springer, Heidelberg (2009)
4. Millan, W., Fuller, J., Dawson, E.: New concepts in evolutionary search for boolean functions in cryptology. *Computational Intelligence* 20(3), 463–474 (2004)
5. Sarkar, P., Maitra, S.: Construction of nonlinear boolean functions with important cryptographic properties. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 485–506. Springer, Heidelberg (2000)
6. Sarkar, P., Maitra, S.: Nonlinearity bounds and constructions of resilient boolean functions. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 515–532. Springer, Heidelberg (2000)
7. Zheng, Y., Zhang, X.-M.: Improved upper bound on the nonlinearity of high order correlation immune functions. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 262–274. Springer, Heidelberg (2001)
8. Pasalic, E., Maitra, S., Johansson, T., Sarkar, P.: New constructions of resilient and correlation immune boolean functions achieving upper bound on nonlinearity. *Electronic Notes in Discrete Mathematics* 6, 158–167 (2001)
9. Aguirre, H., Okazaki, H., Fuwa, Y.: An Evolutionary Multiobjective Approach to Design Highly Non-linear Boolean Functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2007, pp. 749–756 (2007)
10. Clark, J.A., Jacob, J.L., Stepney, S., Maitra, S., Millan, W.: Evolving Boolean Functions Satisfying Multiple Criteria. In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 246–259. Springer, Heidelberg (2002)
11. Picek, S., Jakobovic, D., Golub, M.: Evolving Cryptographically Sound Boolean Functions. In: GECCO (Companion), pp. 191–192 (2013)
12. Adams, C., Tavares, S.: Generating and counting binary bent sequences. *IEEE Transactions on Information Theory* 36(5), 1170–1173 (1990)
13. Jakobovic, D., et al.: Evolutionary computation framework (December 2013), <http://gp.zemris.fer.hr/ecf/>
14. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
15. Yao, X.: Optimization by genetic annealing. In: Proc. of 2nd Australian Conf. on Neural Networks, pp. 94–97 (1991)
16. Fuller, J., Dawson, E., Millan, W.: Evolutionary generation of bent functions for cryptography. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 3, pp. 1655–1661 (December 2003)

A Memetic Algorithm for Multi Layer Hierarchical Ring Network Design^{*}

Christian Schauer and Günther R. Raidl

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
{schauer,raidl}@ads.tuwien.ac.at

Abstract. We address the Multi Layer Hierarchical Ring Network Design Problem. This problem arises in the design of large telecommunication backbones, when high reliability is emphasized. The aim is to connect nodes that are assigned to different layers using a hierarchy of rings of bounded length. We present a memetic algorithm that focuses on clustering the nodes of each layer into disjoint subsets. A decoding procedure is then applied to each genotype for determining a ring connecting all nodes of each cluster. For local improvement we incorporate a previous variable neighborhood search. We experimentally evaluate our memetic algorithm on various graphs and show that it outperforms the sole variable neighborhood search approach in 13 out of 30 instances, while in eight instances they perform on par.

1 Introduction

In this paper we present a memetic algorithm for the *Multi Layer Hierarchical Ring Network Design* (MLHRND) problem. MLHRND arises in the field of telecommunication network design and finds applications in large, hierarchically structured networks with a strong need of survivability. It originates from a cooperation with an Austrian telecommunication provider.

Since our society increasingly depends on large and fast telecommunication networks, the matter of reliability became more and more important. In particular, it has to be avoided that larger parts of the network become disconnected in case of limited failures of devices or links. The simplest way to achieve survivability in a network is the use of a ring topology since the network stays connected in case of a single node or link failure.

For the backbone of wide area networks a single ring would not be efficient anymore. The failure of two nodes or links at the same time could disconnect large parts of the network. Moreover, requirements with respect to bandwidth and maximal delays physically limit the size of a ring. To fulfill physical constraints and ensure a high degree of survivability in larger networks, multiple interconnected rings are frequently used as backbones. To allow scalability this interconnection is often realized in a hierarchical fashion using rings on every layer of the

^{*} This work has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT10-027.

hierarchy. Such a network is hence called a *Hierarchical Ring Network*. If two rings are connected over a single node (*single homing*), the network can compensate for a link failure but does not stay connected if the concatenation node fails. To additionally cover this situation the rings must be connected over two different nodes on each ring, which is also called *dual homing*. The *Multi Layer Hierarchical Ring Network Design* (MLHRND) deals with a hierarchical structure spanning nodes on multiple layers using rings of bounded length and dual homing to ensure fault tolerance in case of single link and node failures. In this work we divide MLHRND into two depending subproblems, a partitioning problem to determine the nodes belonging to each ring and a ring computation problem for each partition. We propose a memetic algorithm for addressing the partitioning aspect, while a decoding procedure determines the actual rings for each partition. Furthermore, we practically evaluate this approach in experiments.

The rest of the paper is organized as follows. Related work is discussed in Section 2, while Section 3 provides a formal definition of the problem. In Section 4 we describe our memetic algorithm. Computational results are presented in Section 5. We conclude this paper in Section 6.

2 Related Work

In [8], we introduced MLHRND for the three-layer case and described a variable neighborhood search (VNS) and a greedy randomized adaptive search procedure (GRASP) for heuristically solving it. We further argued that MLHRND is NP-hard, even for the three layer case, since the classical *Capacitated Vehicle Routing Problem* can be reduced to MLHRND. To our knowledge MLHRND has not been addressed by other authors in the literature yet. However, several other problems are strongly related to it.

Balakrishnan et al. [1] describe the *Multi-Level Network Design* (MLND) problem, a generalization of the well-known Steiner network problem [4]. Nodes of the network are assigned to L different levels. For *Two-Level Network Design* ($L = 2$) the authors discuss two MIP formulations and point out that those approaches can easily be extended to a higher number of levels.

The *Ring Spur Assignment Problem* (RSAP) was introduced by Carroll and McGarraghy in [2]. The authors present a MIP formulation using connectivity constraints and an attempt for a cutting plane approach to solve larger problem instances. In RSAP the network is structured by a *tertiary ring* to which *local rings* connect to. Additionally, some nodes can be connected to a local ring by a single edge, which the authors call a *spur*. Thus, the resulting network is a three level network with rings on the top two levels and spurs on the third.

Gendreau et al. describe in [6] the *Ring Design Problem*, where nodes on a single layer are connected via interconnected rings, and propose an integer programming formulation with a quadratic objective function. The authors argue that heuristic techniques are needed to solve large size instances. Therefore, they present three ring construction heuristics based on TSP heuristics and three destroy and reconstruct approaches for post-optimization.

Proestaki and Sinclair present in [7] a variant of MLND using rings and dual homing for matters of survivability, where the node to level assignment is not given a priori but to determine during the optimization process. The objective function incorporates both the traffic on the rings and the overall ring length. As an exact approach the authors present a binary integer linear programming formulation. Additionally, they discuss a partition, construct, and perturb heuristic that iteratively, for each level, creates a solution.

Similarities further exist between MLHRND and some variations of capacitated vehicle routing problems when considering satellite depots. For instance Schwengerer et al. [9] study the *Two-Echelon Location-Routing Problem*, a combination of the VRP and the Facility Location Problem (FLP). To solve the problem the authors present a variable neighborhood search. As in MLHRND the node set is split into three subsets, which are platforms (layer 1), satellites (layer 2), and customers (layer 3). In the context of vehicle routing dual homing is not a meaningful requirement.

3 Multi Layer Hierarchical Ring Network Design

This section gives a formal definition of MLHRND and discusses some observations concerning this problem.

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E . A weighting function assigns costs $c_{ij} \geq 0$ to each edge $(i, j) \in E$. Moreover, V is partitioned into $K \geq 3$ disjoint subsets V_1, \dots, V_K representing the layers each node belongs to. Edges exist between all pairs of nodes of the same and the successive layer, i.e., $E = \bigcup_{k=1, \dots, K} (V_k \times V_k) \cup \bigcup_{k=1, \dots, K-1} (V_k \times V_{k+1})$.

A feasible solution to MLHRND is a subgraph $G_L = (V, E_L)$ connecting all nodes in V and satisfying the following conditions; see Figure 1 for an example.

1. All nodes in V_1 are connected by a single independent ring containing no other node.
2. The remaining layers are connected by $K - 1$ respective sets of paths containing no nodes from other layers. Each node must appear in exactly one path, i.e., the paths are node and edge disjoint to ensure reliability.
3. The end nodes of each path at layer $k \in \{2, \dots, K\}$ are further connected to two different nodes (*hubs*) in layer $k - 1$, i.e., dual homing is realized. We refer to the edges connecting paths to hubs as *uplinks*.
4. The two hub nodes, a path is connected to, must themselves be connected by a simple path at their layer, i.e., the connection to a ring may not be established via more than two layers.
5. The lengths of layer $k \in \{2, \dots, K\}$ paths in terms of the number of edges is bounded below and above by $b_k^l \geq 1$ and $b_k^u \geq b_k^l$, respectively.

The objective is to find a feasible solution with minimum total costs:

$$c(E_L) = \sum_{(i,j) \in E_L} c_{ij}$$

From condition 1 we can conclude that finding the layer 1 ring resembles the classical Traveling Salesman Problem (TSP). Since there are no further limitations for the layer 1 ring, this subproblem can be solved independently.

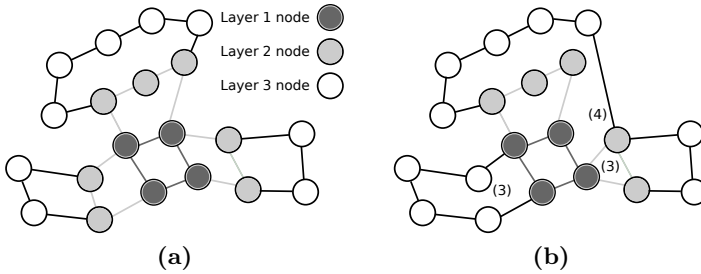


Fig. 1. Schematic representations for $K = 3$, of (a) a feasible solution and (b) an infeasible solution (the numbers in brackets indicate the violated constraints)

The situation changes when considering the remaining layers. Through the combination of the dual homing aspect (condition 3) and the connection of the hubs via simple paths (condition 4) the optimal structures of layers $k \in \{2, \dots, K\}$ strongly depend on each other and cannot be treated separately. This combination marks the challenging aspect of MLHRND both to model and solve it. Note that relaxing either condition 3 or condition 4, each layer could be solved independently to achieve an overall optimal solution.

4 A Memetic Algorithm for MLHRND

As already mentioned, layer 1 corresponds to the classical TSP and can be solved independently. As the TSP is well studied and our primary interest lies in the structurally more complex further layers, we use in our experiments the Concorde TSP solver [3] to determine an optimal layer 1 ring. The following memetic algorithm (MA) is therefore used to solve the remaining layers. The main idea behind the MA is to split MLHRND into two subproblems: the first, to cluster the nodes of each layer into different subsets; the second, to compute Hamiltonian paths through the subsets and determine the uplinks for the paths in order to form together with the upper layer feasible rings. While the MA is used to optimize the clusters, a decoding procedure is used for calculating the paths and rings.

4.1 Representation and Decoding Procedures

A main problem, when genetic algorithms (GAs) are applied to clustering problems, is to find a proper encoding scheme for the genotype that is independent from the order of the clusters and the order of the elements within each cluster of the phenotype. An encoding scheme that compensates these problems is *linear linkage encoding* (LLE) proposed by Du et al. [5]. This representation stores for each gene the index of a fellow gene within the same cluster. By requesting that the stored index must be greater or equal than the own index, LLE provides a unique representation for each individual without the mentioned encoding problem. For MLHRND we label all nodes with indices $1, \dots, |V|$ to define a natural

order on V . By sorting the nodes within each cluster and the clusters according to the first node of each cluster, both in ascending order, we achieve a unique representation of a candidate solution similar to LLE. Our tests showed that the computational effort for this encoding process is negligible.

Since finding a minimum weight Hamiltonian path is an NP-complete task, we implemented a heuristic decoding procedure. The idea is to first compute the path through the nodes of a cluster and in a second step determine the uplinks for the end nodes of the path. Preliminary tests showed that a decoding procedure based on the nearest neighbor principle for the TSP performed best in terms of solution quality and runtime. This procedure uses the shortest edge within a cluster as initial path, which is then iteratively extended by greedily appending nodes on both ends. The next node to be appended is always the nearest one to one of the end nodes. Ties are broken in favor of the node with the smaller index to keep the heuristic deterministic. This procedure is repeated until all nodes within a cluster are connected to a Hamiltonian path. Moreover, tests showed that improving each path with local search using a two edge exchange neighborhood structure pays off considering the increase of runtime and the improvement potential. To additionally speed up the decoding we store all decoded partitions with their resulting paths in an archive for later reuse.

Moreover, we designed a decoding strategy based on integer linear programming techniques to optimally determine the Hamiltonian path together with the uplinks. This approach allows an optimal decoding of the genotype but it performs too slowly to decode every partitioning created by the MA. Instead we use it only in the end to exactly decode the best solution found by the MA.

4.2 Initial Population

For the initial population we create one third of individuals using the randomized construction heuristic we proposed in [8]. This heuristic appends a path based on the nearest neighbor idea until b_k^u is reached and then starts a new path, which results in solutions with few but long paths. For this work we adapted this heuristic by also considering hub nodes and therefore to close a path in an earlier stage in case a hub node is nearer than a node on the same layer. Another third of the individuals is generated by a version of this heuristic, randomized in the same way as the first one, which creates solutions with more and shorter paths. Furthermore, we implemented another construction heuristic based on the savings principle. This heuristic iteratively creates paths for each layer by computing the savings for every pair of nodes on the layer and sorting them in descending order. Then we iterate over the savings until all nodes of the layer are connected to feasible paths, which results in solutions with many but short paths. By randomly shuffling instead of sorting the positive savings on layer 2 we obtain a randomized construction heuristic to create the remaining individuals. For the remaining layers we use the sorted savings lists, otherwise the solutions obtained would be too random. Using these three heuristics with their different solution characteristics leads to a promising and diverse initial population.

4.3 Recombination and Mutation

We designed two crossover operators using two parents and creating two descendants. Both operators recombine on the cluster level employing the same strategy for each layer $k \in \{2, \dots, K\}$.

One Point Crossover (1PX) chooses a random splitting point between the list of clusters on each layer. In a first step, the clusters before the splitting point are copied from parent one to offspring one and from parent two to offspring two, respectively. Afterwards, the clusters after the splitting point are passed vice versa from parent two to offspring one and from parent one to offspring two, respectively. Now care must be taken that none of the nodes occurs twice in different clusters. Therefore, a check is needed for each node whether it can be added to the cluster or must be dropped. If too many nodes are dropped and b_k^l is not satisfied, the cluster is not added to the offspring at all. This means on the other hand that some nodes might be skipped. Thus, in a third step remaining nodes are iteratively added to clusters containing nodes located nearby and not fulfilling b_k^l . The idea is that the increase of length of the Hamiltonian path might be smaller if there are other nodes closely located to the new one. If all clusters have size b_k^u , then a new cluster is created and the node is added there.

Uniform Crossover (UX) decides randomly from which parent the next cluster is passed to the offspring. This means that the next cluster can either be passed from parent one or two to offspring one and correspondingly either from parent two or one to offspring two. If the number of clusters differs for the parents, the remaining clusters are added to both descendants. As in 1PX clusters not satisfying b_k^l are not added to the offspring at all. Again care must be taken that none of the nodes is added twice to an offspring. To add remaining nodes UX uses the same insertion strategy as 1PX.

The mutation operators are based on neighborhood structures described in Section 4.4. A mutation of an individual resembles a single random move in the respective neighborhood.

Two Node Exchange Mutation (2NE-M) swaps two nodes from different clusters on the same layer. At first, a layer, two clusters on this layer, and two nodes on the respective clusters are determined randomly. These two nodes are then swapped between the two clusters. Since no constraints could be violated this mutation is always applicable without further restrictions.

One Node Move Mutation (1NM-M) shifts a single node from one cluster to another on the same layer. The layer, the two clusters, and the shifted node are all determined randomly and then the mutation is performed. For 1NM-M care must be taken that neither b_k^l for the original cluster nor b_k^u for the destination cluster are violated. If no such clusters exist on the chosen layer then the mutation is automatically tried on another layer.

Merge Cluster Mutation (MC-M) merges two clusters on the same layer. The layer and the two clusters are chosen randomly. The two clusters must be selected so that after the merge b_k^u is not exceeded. In case a merge is not possible on the chosen layer an attempt on another layer is made.

Split Path Mutation (SP-M) operates on a decoded path of the phenotype and splits this path at a given position into two new paths. Since the nodes within a cluster in the genotype are stored in ascending order, dividing a cluster into two would not be meaningful. The layer and the splitting point on the path are determined randomly ensuring that the two new paths both exceed b_k^l . Again if a split is not possible on the chosen layer another layer is tried.

4.4 Local Improvement by Variable Neighborhood Search

For local improvement we use a downgraded version of the general variable neighborhood search (VNS) proposed in [8]. It includes an embedded variable neighborhood descent (VND), in which we consider the following neighborhood structures:

Two Edge Exchange (2EE) is applied to every path on every layer separately starting with the first uplink and ending with the second investigating all feasible candidate solutions that differ in at most two edges.

Change Uplinks (CU) finds the optimal uplinks for each path considering that the hub nodes must be different and connected via a simple path.

Split Rings (SR) divides a long path into two feasible shorter paths and connects them to the preceding layer.

Two Node Exchange (2NE) swaps two nodes from different paths on the same layer.

One Node Move (1NM) shifts a single node from one path to another on the same layer taking care that neither b_k^l on the original path nor b_k^u on the destination path is violated.

Append Rings (AR) can be seen as inversion of SR and connects the end nodes of two short paths to a feasible long path.

We did not use the *Three Edge Exchange (3EE)* and *Merge Rings (MR)* neighborhood structures from [8] because tests showed that their application was time consuming but had only a minor impact on the solution quality.

Taking a closer look one can observe that a move within 2EE and CU only affects a single path while a move in the other neighborhood structures affects greater parts of the solution. Therefore, we separated 2EE and CU from the others in an intra-VND, which is called every time a move within one of the other neighborhood structures was performed and only applied to the affected paths. The operators in the main VND are applied in the order as listed before.

For shaking we used the same strategy as in [8], i.e., $2K - 2$ shaking neighborhood structures $\mathcal{N}_{1, \dots, 2K-2}$ defined as follows:

$$\begin{aligned} \mathcal{N}_i &= \text{one random move in 2NE on layer } K - i + 1, & \forall i = 1, \dots, K - 1 \\ \mathcal{N}_i &= \text{one random move in 1NM on layer } 2K - i, & \forall i = K, \dots, 2K - 2 \end{aligned}$$

Since the VNS operates on the solution graph the shaking neighborhood structures cannot be compared with the mutation operators, which permute the clusters. Therefore, it makes sense to not only apply the VND but also to perform

shaking within the local improvement phase. In this case, we terminate the VNS if no improvement after shaking in \mathcal{N}_{2K-2} was found. Therefore, the increase in runtime pays off in relation to the improvement obtained.

5 Results

For testing purposes we focus on the $K = 3$ layer scenario and use the same TSPLIB¹ based benchmark instances as in [8] for the VNS and GRASP. Additionally, we extended the test set by new random instances especially to increase the variety of smaller instances. For this purpose, we randomly placed nodes in a grid of size 10000×10000 , used k -means clustering to determine the layer 1 and layer 2 nodes, and added corresponding edges. In this way we obtained a total of 74 test instances with up to 439 nodes. By using different combinations of upper bounds b_k^u for the path lengths we obtained 380 test cases. For the lower bound b_k^l we always assumed one edge as the minimum length for all paths. All these test instances can be downloaded from www.ads.tuwien.ac.at/w/Research/Problem_Instances.

We tested our MA using either 1PX or UX as crossover operators. As parameter setting for all test cases we used a population size of 50 individuals, tournament selection over three individuals and elitism for the best five. As stopping criterion we used a time limit as indicated in Table 1. We did not allow duplicates within the population of the same generation. The mutation rate was set to 9% for 2NE-M, 1NM-M, and MC-M and 4.5% for SP-M. We performed local improvement on the best five individuals every 500 generations. For instances with less than 300 nodes we used the VNS for larger instances the VND only, since the increase in runtime was too high. In the end, the best solution found by the MA is always improved by VNS and finally the exact decoding procedure is applied. We implemented our approach in Java 1.6 using IBM CPLEX 12.6 for the exact decoding procedure. For each of the test cases we performed 30 runs executed on a single core of an Intel Xeon (Nehalem) Quadcore CPU with 2.53GHz and 3GB of RAM. For comparison we adapted the VNS from [8] by using the intra-VND (with 3EE on the second position) with the main VND (with MR on the last position) and executed this approach on all test cases.

Results are summarized in Table 1. The columns have the following meaning: *instance* indicates the underlying graph our derived test cases are based on (with the number of nodes in the graph at the end of the name); # denotes the number of different test cases for each graph; b_2^u and b_3^u lists the different upper layer bounds, where all combinations were tested; t refers to the runtime limit in seconds; for the MA with 1PX and UX crossover and VNS the average objective values (*score*) together with their standard deviations (*dev*) are listed; columns p_{AB} show a statistical comparison between approach A and B based on a Student's t-test with an error level of 5%, $<$ ($>$) means that A performs better(worse) than B , \approx indicates no significant difference between A and B .

¹ <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>

Table 1. Experimental results for the MA using either 1PX or UX as crossover operator and the VNS with average objective values and their standard deviations. Columns p_{AB} show a statistical comparison between approach A and B based on a Student’s t-test with an error level of 5%. The first columns list the instance name, the number of test cases, test values for b_2^u and b_3^u , and the runtime limit for each instance.

instance	#	b_2^u	b_3^u	t[s]	1PX		$p_{1PX/UX}$	UX		$p_{UX/VNS}$	VNS	
					score	dev		score	dev		score	dev
ulysses22	2	4	4,6	150	123.31	0.28	> ≈	123.20	0.24	<	123.36	0.27
rand25	2	4	4,6	150	71255.47	221.34	≈ <	71255.47	221.34	<	72820.93	801.01
rand30	4	4	4,6	150	88484.53	3542.46	≈ <	88365.69	3422.10	<	89373.55	2975.65
rand35	4	4	4,6	150	89182.22	2888.34	≈ <	89167.36	2874.31	≈	89457.32	3067.79
rand45	4	4	4,6	150	99145.13	2039.62	≈ <	99126.84	1992.77	<	101496.14	3640.91
att48	4	4	4,6	150	59663.81	632.16	≈ <	59596.17	681.46	<	60169.25	966.66
eil51	4	4	4,6	150	743.72	15.56	≈ <	743.02	15.63	≈	746.89	16.28
berlin52	4	4	4,6	150	13370.03	201.04	≈ <	13364.96	197.50	<	13451.39	279.67
rand55	4	4	4,6	150	111846.88	2971.71	≈ <	111882.44	2878.37	<	113360.76	2299.75
rand70	12	4,7	4,7,11	150	126402.60	2322.23	> <	125926.40	2264.46	<	127383.39	3835.20
eil76	12	4,7	4,7,11	150	902.71	21.82	≈ <	902.03	23.09	≈	903.94	24.27
rand85	18	4,7	4,7,11	150	136312.04	3237.14	≈ <	136050.13	3227.32	<	136995.32	3755.33
gr96	18	4,7	4,7,11	300	925.80	21.40	≈ <	926.07	22.21	<	930.99	21.11
kroA100	18	4,7	4,7,11	300	40043.51	1132.73	≈ <	39992.35	1151.43	≈	40066.80	1152.69
kroB100	18	4,7	4,7,11	300	40759.91	1068.90	≈ <	40780.24	1122.37	≈	40826.96	1219.67
bier127	12	7,11	7,11,14	300	202817.43	2601.11	≈ >	202674.00	2836.98	>	201500.32	2445.40
ch150	18	7,11	7,11,14	300	11719.20	236.62	≈ >	11715.27	246.43	>	11638.03	230.24
rand175	18	7,11	7,11,14	300	184434.64	4073.46	≈ >	184749.36	4161.79	>	182736.31	3843.28
kroA200	18	7,11	7,11,14	300	53045.74	986.66	<	53183.41	1089.88	>	52854.56	1018.72
kroB200	18	7,11	7,11,14	300	53148.67	1024.49	≈ >	53229.85	1023.96	>	52984.27	1020.04
gr229	12	11,16	11,16,19	600	2840.61	66.71	≈ >	2842.48	65.43	>	2821.13	71.97
rand250	12	11,16	11,16,19	600	214841.28	2983.39	≈ >	215245.46	3347.49	>	212409.05	2223.95
rand275	18	11,16	11,16,19	600	219517.82	4488.71	<	220505.76	4672.04	>	217118.47	3608.25
pr299	18	11,16	11,16,19	600	88667.95	1264.57	<	88920.18	1383.50	>	87873.12	1033.80
lin318	18	11,16	11,16,19	600	77173.81	1718.17	<	77612.38	1835.81	>	77019.39	1264.34
rand350	18	11,16	11,16,19	600	248766.28	5152.62	<	250009.87	5706.74	≈	250113.76	4810.65
rand375	18	11,16	11,16,19	600	259471.79	5313.26	<	261068.47	6000.87	>	259992.72	5100.66
rand400	18	11,16	11,16,19	900	269699.91	5854.83	<	271924.26	6542.30	>	269349.24	5199.97
gr431	18	11,16	11,16,19	900	3373.65	57.92	<	3401.35	63.94	<	3455.16	65.91
pr439	18	11,16	11,16,19	900	201790.51	6992.98	<	204776.11	7414.42	<	206108.42	8830.63

Firstly, we observe that by using our new VND approach we are able to improve the VNS results significantly compared to [8]. The GA without local improvement cannot compete with the VNS except for rand25. Therefore and due to space limitations, we do not present the results for the GA alone. Of greater interest is the comparison between the MA and VNS. For all 15 instances with up to 100 nodes the MA with both crossover operators outperforms the VNS in terms of solution quality, in ten cases the difference is significant (see columns p_{AB}). In many cases UX delivers the best results. For instances of this size the MA can cover a large search space, while the VNS gets stuck in a local optimum too early. The situation changes for instances with 127–300 nodes, where VNS always performs significantly better. Here the VNS can show its strength and cover a larger search space because the application of VND is still fast and many iterations are performed. For more nodes the MA, especially with 1PX, performs again better than the VNS except for lin318 and rand400, where the VNS is not significantly better, and for rand375, where 1PX is not significantly better.

As already mentioned the application of VND becomes very time consuming for larger instances and only some iterations for the VNS are possible in the given time limit. The MA still can compute at least 3000 generations especially with 1PX, which in practice is faster than UX. Therefore, we conclude that especially for larger instances the MA is the algorithm of choice.

6 Conclusions and Future Work

We presented a memetic algorithm to solve the Multi Layer Hierarchical Ring Network Design problem. The basic concept is to use the MA to cluster the nodes of each layer into disjoint subsets, while a decoding procedure computes a Hamiltonian path through each cluster and finds uplinks to determine a feasible solution. The approach includes a variant of a previously published VNS with an embedded VND. The VND has been adapted by distinguishing between neighborhood structures that work on the path level or on the whole solution. In this way we were able to speedup the VND significantly and also obtained better results for the VNS alone. The MA outperforms the VNS on instances with up to 100 and more than 350 nodes. In the future we will focus on decomposition techniques, e.g., Benders decomposition for solving medium sized instances exactly, and the design of large neighborhood structures.

References

1. Balakrishnan, A., Magnanti, T.L., Mirchandani, P.: The Multi-level Network Design Problem. Tech. Rep. 3366-91, Massachusetts Institute of Technology (1991)
2. Carroll, P., McGarraghy, S.: Investigation of the ring spur assignment problem. In: Bigi, G., Frangioni, A., Scutellà, M. (eds.) Proceedings of the 4th International Network Optimization Conference (INOC 2009). pp. MB1–3 (2009)
3. Cook, W.J.: Concorde TSP Solver, <http://www.math.uwaterloo.ca/tsp/concorde/> (accessed: March 13, 2014)
4. Dreyfus, S., Wagner, R.A.: The Steiner Problem in Graphs. *Networks* 1, 195–207 (1972)
5. Du, J., Korkmaz, E., Alhajj, R., Barker, K.: Novel clustering approach that employs genetic algorithm with new representation scheme and multiple objectives. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 219–228. Springer, Heidelberg (2004)
6. Gendreau, M., Labbé, M., Laporte, G.: Efficient heuristics for the design of ring networks. *Telecommunication Systems* 4(1), 177–188 (1995)
7. Proestaki, A., Sinclair, M.: Design and dimensioning of dual-homing hierarchical multi-ring networks. *IEE Proceedings Communications* 147(2), 96–104 (2000)
8. Schauer, C., Raidl, G.R.: Variable Neighborhood Search and GRASP for Three-Layer Hierarchical Ring Network Design. In: Coello Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 458–467. Springer, Heidelberg (2012)
9. Schwengerer, M., Pirkwieser, S., Raidl, G.R.: A variable neighborhood search approach for the two-echelon location-routing problem. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012. LNCS, vol. 7245, pp. 13–24. Springer, Heidelberg (2012)

Scheduling the English Football League with a Multi-objective Evolutionary Algorithm

Lyndon While^{1,*} and Graham Kendall²

¹ Computer Science & Software Engineering, The University of Western Australia

² ASAP Research Group, School of Computer Science, University of Nottingham Malaysia Campus, Malaysia and University of Nottingham, UK
lyndon.while@uwa.edu.au, graham.kendall@nottingham.edu.my

Abstract. We describe a multi-objective evolutionary algorithm that derives schedules for the English Football League over the busy New Year period according to seven objectives. The two principal objectives are to minimise travel distances for teams and supporters, and to minimise so-called “pair clashes” where teams which are geographically close play at home simultaneously, which can cause problems for police, and other logistical issues. The other five objectives implement various problem constraints. The schedules derived are often superior both to those used in the relevant years, and to those previously published in the literature, especially for the harder problem instances. In addition, the system returns a set of schedules offering different trade-offs between the main objectives, any of which might be of interest to the authorities.

Keywords: Sports scheduling, Multi-objective evolutionary algorithms.

1 Introduction

The English Football League is structured around a promotion/relegation system where teams can move up or down divisions, depending on their performances. At the top of the league structure is the English Premier League (EPL), which includes famous teams such as Arsenal, Chelsea, Liverpool, and Manchester United. Beneath the EPL there are three other leagues: the Championship, League 1, and League 2¹. These four divisions represent the top level of English football.

The EPL has 20 teams and the other three divisions have 24 teams each, a total of 92 teams. Each division is a double round robin tournament, where teams play each other twice, once at each venue. This requires $20 \times 19 = 380$ matches in the EPL, and $24 \times 23 = 552$ matches in each of the others. Thus there are 2,036 fixtures to be scheduled in any one season, a significant task.

It might be assumed that each division can be scheduled individually, as no team from one division plays against a team from another division. However, dependencies exist between the divisions. The main dependency is around

* Corresponding author.

¹ The actual names change, depending on sponsorship arrangements.

pairings. Teams that are paired should (ideally) not play at home on the same day, to reduce the burden on police and other infrastructure. For example, Manchester United and Manchester City are paired as the police would prefer them not to both be playing at home on the same day. Pairings exist across divisions: e.g. Liverpool, Everton, and Tranmere Rovers are all paired.

We focus on scheduling the English Football League over the busy Christmas and New Year period, when each team plays either two or four matches in a short time, and the constraints on the schedule are more difficult than usual, e.g. bad weather, police leave, and clashes with other holiday commitments for supporters. We try to minimise the distance traveled by supporters, which was the primary focus of our previous work[1,2,3]. We would also like to minimise the number of *pair clashes* where paired teams play at home on the same day. Minimising pair clashes and minimising travel are conflicting objectives and it seems appropriate to investigate a multi-objective approach for this problem. Previous attempts[3] have used multiple runs of a single-objective algorithm.

The principal contribution of this paper is the description and analysis of a multi-objective evolutionary algorithm that derives schedules over the Christmas and New Year period. The algorithm minimises the total travel distance (to save supporters from having long journeys), and it also minimises the number of pair clashes (to save policing resource). Five constraint objectives are also managed: these are presented in Section 3, and a more formal problem description is given in [1]. The schedules presented are superior both to those used by the football authorities, and also to those previously published in the literature, especially for the larger problem instances. In addition, the system returns a set of schedules offering different trade-offs between the two objectives, any of which might be of interest to the authorities.

The rest of the paper is structured as follows. Section 2 describes relevant background material and previous work, and Section 3 describes the problem that we address. Section 4 describes the details of our multi-objective approach, and Section 5 describes the results achieved. Section 6 concludes the paper.

2 Background

2.1 Multi-objective Approaches to Sports Scheduling

Multi-objective approaches to sports scheduling have received limited attention in the literature. This is perhaps surprising, given the multi-objective nature of this sector, e.g. maximise crowd attendance, minimise costs, minimise policing, maximise sales, minimise travel, maximise the gap between return matches, etc.

Evans[5], as far back as 1988, proposes a decision support system for scheduling umpires in baseball. One objective is to minimise the travel costs of umpires, with the other objective being concerned with the number of times an umpire can officiate over a given team.

Duarte and Ribeiro[4] consider a bi-objective referee assignment problem, where they minimise the difference between the target and assigned games for each referee, and minimise the idle time between games for each referee.

Barone *et al.*[6] investigate the Australian Football League wrt a number of competing objectives: balancing the number of home games for each team, balancing the sequence of home/away games (as too many consecutive games at home (resp. away) is viewed as unfair), balancing the effects of travel in a large country, maximising profit, and distributing the games across the country on a given weekend. Their multi-objective evolutionary algorithm was able to produce good results which offer a range of options to the client.

While and Barone[7] focus on Super 14 rugby. This tournament takes place in several countries (Australia, New Zealand, and South Africa), distributed over twelve time zones. They balance the number of home games a team has, whilst optimising travel requirements and distributing the games in each round evenly across the three countries, to optimise the use of prime time TV spots. Again they use a multi-objective evolutionary algorithm, producing results that offer significant improvements over the fixtures adopted by the organising body.

Craig *et al.*[8] use a multi-objective evolutionary algorithm to investigate the National Hockey League. The objectives are minimising travel, providing equity in rest time between games, and minimising consecutive home/away game sequences. The results are superior to the 2008–09 schedules that were used, and they offer a range of trade-offs due to the nature of the multi-objective algorithm.

Kendall and Westphal[2] use CPLEX in an attempt to solve the problem of scheduling football matches over the New Year period whilst trying to minimise the total distance travelled. From a base model they vary the parameters in order to find solutions more acceptable to the individual clubs, e.g. by limiting the maximum distance that a single team would travel. A variety of experiments are presented to demonstrate the effectiveness of this approach.

Kendall *et al.*[3] again explore the idea of using multiple runs with varying parameters. They minimise both the distance travelled and the number of pair clashes in an attempt to find good trade-offs. The current paper tackles the same problem as [3], but using a multi-objective approach. The aim is to get better solutions in a single run, both saving time and offering the client superior solutions.

Survey papers related to sports scheduling are available in [9,10].

2.2 Multi-objective Optimisation

In a multi-objective optimisation problem, potential solutions are assessed according to two or more independent quantities. The characteristic of good solutions is that improving in one objective can be achieved only by worsening in at least one other objective. An algorithm for solving such problems returns a set of solutions offering different trade-offs between the various objectives.

Consider a problem where the fitness function maps a solution x into a fitness vector \overline{f}_x . A solution x *dominates* a solution y iff \overline{f}_x is at least as good as \overline{f}_y in every objective, and is better in at least one objective. x is *non-dominated* wrt a set of solutions X iff there is no solution in X that dominates x . X is a *non-dominated set* iff every solution in X is non-dominated wrt X . The set of fitness vectors corresponding to a non-dominated set is a *non-dominated front*.

A solution x is *Pareto optimal* iff x is non-dominated wrt the set of all possible solutions. Such a solution is characterised by the fact that improvement in one objective comes only at the expense of other objectives. The *Pareto optimal set* is the set of all Pareto optimal solutions. Multi-objective optimisation aims to find (or approximate) this Pareto optimal set.

With multiple objectives there is only a partial order on solutions, which causes problems for selection in an evolutionary algorithm. The usual solution is to define a ranking on solutions: one popular scheme[11] defines the *rank* of a solution x wrt a set X to be the number of solutions in X that dominate x . Selection is then based on ranks: a lower rank implies a better solution.

Precise definitions of all these terms can be found in [12].

3 Problem Statement

As stated in Section 1, the top four English divisions contain 92 teams, and require 2,036 fixtures to be scheduled each season. Over Christmas and New Year, each team plays either two or four matches. One (resp. two) match has to be played at home, and the other (resp. two) has to be played at an away venue. This time of the year is one of the few times when all teams have to play. It may seem that scheduling part of a season might just create difficulties for other dates, but in these competitions many games are moved each year for a variety of reasons, so it is common for the authorities to focus their efforts in this way.

The full problem can be defined as follows. Further details are available in [1]. Given a set of teams allocated to the four divisions in a given year, given the distance between each pair of teams, and given for each team a set of “pairs”:

1. There are either two or four rounds (this depends on the season, but following [2] we always generate a two-round fixture and a four-round fixture).
2. Each team plays one game in each round against another team in its division.
3. Each team alternates home and away games.
4. No team can play another team more than once.
5. No team can play against any of its pairs (this constraint is sometimes violated by the football authorities, but we enforce it).
6. Only six London teams can play at home in any round.
7. Only three London EPL teams can play at home in any round.
8. Only four Greater Manchester teams can play at home in any round.

A schedule is measured against two objectives:

- The total distance travelled by the away teams across all rounds.
- The total number of pair clashes (where paired teams play at home on the same day) across all rounds. For the first six seasons we study, the minimum possible clashes for the two-round case is eight². In 2008–09, Mansfield Town were demoted from the league, reducing the minimum to seven.

² Three mutually-clashing teams implies at least one clash/two rounds (six occurrences of this), four mutually-clashing teams implies at least two clashes/two rounds (one occurrence of this). Note that the number of pair clashes for four rounds is doubled, because the home teams in Round k are the same in Round $k + 2$.

For each scenario, a target number of clashes is given based on the schedule actually used in that year[1,2].

4 Methodology

This section describes the details of our algorithm: the genetic representation and variation operators used, the seven objectives and their quantification, and details of initialisation, termination, and archiving.

4.1 Representation

Given a problem instance with k rounds, for a division with n teams we have

- a permutation of the teams in that division;
- $k - 1$ permutations of the numbers $0 \dots n/2 - 1$.

The translation from genotype to phenotype (i.e. the schedule) is best explained via a worked example. With four rounds and a single division containing the six teams A,B,C,D,E,F, the genotype

EDAFBC - 201 - 120 - 021

represents the schedule

Round 1: E vs F, D vs B, A vs C
 Round 2: F vs A, B vs E, C vs D
 Round 3: E vs B, D vs C, A vs F
 Round 4: F vs E, B vs A, C vs D

In Rounds 1 and 3, the teams in the first half of the main permutation (i.e. E, D, A) play at home: in Rounds 2 and 4, the teams in the second half of the main permutation (i.e. F, B, C) play at home. In Round 1, the away teams are allocated directly from the second half of the permutation: in Round $j + 1$, the j^{th} list of indices is used to permute the away teams.

Note that the genotype for a particular problem instance will contain one of these combinations for each division in the league. For representation purposes, the divisions can be treated entirely separately.

The representation enforces directly the number of rounds scheduled, and the requirements that each team plays once per round and that teams alternate home and away games (i.e. 1–3 from Section 3). The variation operators described in Section 4.2 maintain these requirements. The other requirements (i.e. 4–8 from Section 3) are enforced via so-called “constraint objectives” in the fitness calculations.

4.2 Variation Operators

Mutation involves randomly selecting either 1 or 2 of the permutations in the genotype, and in each one separately, swapping two of its entries. This mutation is guaranteed to maintain feasibility in the genotype.

No crossover is currently used.

4.3 Objectives

We implement seven objectives. The first five are constraint objectives where a value that is too high indicates an infeasible solution[13,14,15].

1. **Repeats:** Across all rounds, we count the number of games that are repeated (i.e. where t_i plays t_j twice): none are allowed.
2. **Derbies:** Across all rounds, we count the number of games between paired teams: none are allowed.
3. **London EPL:** In each round, we count the number of EPL games played in London: we allow three/round.
4. **London:** In each round, we count the number of games played in London: we allow six/round.
5. **GMR:** In each round, we count the number of games played in Greater Manchester: we allow four/round.
6. **Pair clashes:** In each round, we count the number of times that a team and one of its pairs are both playing at home, and we sum these values for all rounds: we allow a total that depends on the year, from Table 33 of [2].
7. **Distance:** Across all rounds, we sum the distances travelled by the away teams in all games.

Note that the limits set for London EPL, London, and GMR are the minimum values possible given the number of teams in each region and division.

We have tried various ways of combining the five constraint objectives, but we have found that maintaining them as separate objectives gives the best diversity in the population.

4.4 Other Algorithm Details

Selection: We use standard Pareto ranking[11] in selection, and where we need to break ranks, we favour the constraint objectives over the “real” objectives, in the order described in Section 4.3. Thus feasible solutions are favoured in selection, which will tend to increase the proportion of such solutions over time.

Archiving: All non-dominated feasible solutions with distinct fitnesses are archived and are presented in the results of the algorithm. This approach works well because of the discrete nature of pair-clash-counts: the archive will contain only a small part of the population, never more than about twenty solutions.

Initialisation: For the initial set of solutions, we simply choose each permutation randomly.

Termination: The algorithm runs until the feasible solution with minimum pair clashes ceases to show improvement. For the experiments described in Section 5, this typically took 10–20,000 generations.

5 Results

We tested our algorithm on fourteen problem instances. We used the seven seasons 2002–03 to 2008–09 inclusive, and for each season we ran the algorithm to

Table 1. For the 2- and 4-round cases, we give the lower bounds from [2]; the distances for [2] and for the current algorithm; the percentage differences between them; and the run-times. Negative percentages indicate improvements for our proposed algorithm. Each result is from one run with a population of 1,500, roughly 10–20,000 generations.

Two rounds					
	bound	[2]	current	improv.	minutes
2002–3	4,692	4,801	4,825	0.5%	533
2003–4	5,186	5,209	5,244	0.7%	500
2004–5	5,135	5,161	5,216	1.1%	263
2005–6	5,038	5,038	5,067	0.6%	398
2006–7	5,295	5,308	5,374	1.2%	272
2007–8	5,020	5,034	5,076	0.8%	180
2008–9	5,243	5,244	5,259	0.3%	190
Four rounds					
	bound	[2]	current	improv.	minutes
2002–3	11,377	13,813	13,773	–0.3%	643
2003–4	11,896	13,966	13,870	–0.7%	483
2004–5	12,040	13,605	13,576	–0.2%	298
2005–6	12,221	13,785	13,612	–1.3%	328
2006–7	12,409	14,262	14,076	–1.3%	652
2007–8	11,985	14,089	13,897	–1.4%	440
2008–9	12,283	14,671	14,529	–1.0%	501

produce two and four rounds of fixtures. This is consistent with [2] and enables us to compare our results directly. We also compare results with [3], but this paper produced only two rounds of fixtures.

Table 1 summarises the best distance results achieved for results that respect all of the problem constraints, including the same limits on pair clashes as used in [2]. Our runtimes are in line with those used in [2], accounting for the different computer configurations and bearing in mind that [2] did many runs for each instance, whereas our multi-objective approach requires only a single run. Clearly if this methodology were used in the real world, the run times would be acceptable given that it is a once-a-year operation and that the schedules are important from many perspectives, including financial.

It is clear that our multi-objective algorithm performs slightly worse than [2] on the 2-round fixtures where they get very close to the optima, and slightly better on the more complex 4-round fixtures. It is fairly typical of evolutionary approaches to perform (relatively) better on harder problems.

Figure 1 shows the complete results on the 2-round fixtures. The y -axis shows the difference to the distances reported for [2] in Table 1, and the x -axis shows the number of pair clashes, relative to the numbers that were used in the fixtures that were published by the football authorities. That is, a value of zero means that we have the same number of pair clashes, and a negative (resp. positive) value means that we have generated a schedule with fewer (resp. more) pair clashes than the published fixture. Thus the result for [2] sits at (0, 0) for every line. It is not surprising that, as we allow more pair clashes, we have less distance

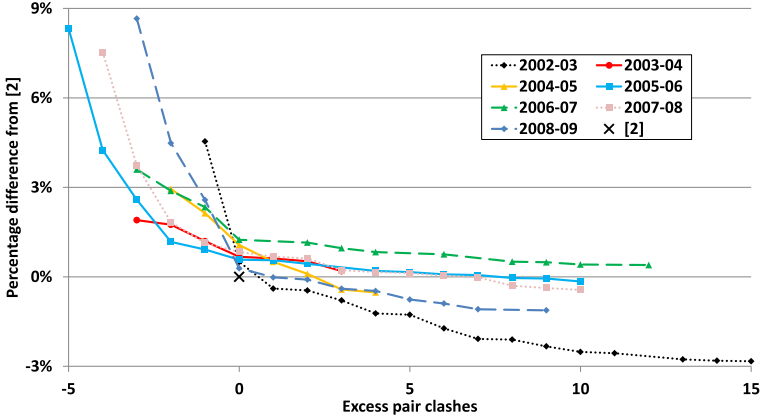


Fig. 1. For the 2-round case, each line plots the best distance result achieved with the corresponding number of pair clashes in excess of the [2] limits, for the same runs as Table 1 plus another set of runs that tried to minimise clashes as far as possible. Negative values indicate reduced distances travelled. The leftmost point on every line has the smallest number of clashes possible, usually eight (but seven for 2008–09).

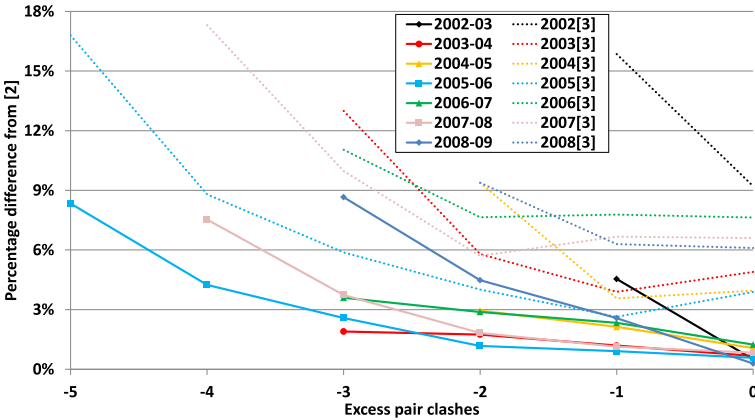


Fig. 2. For the 2-round case, each solid line plots the same data as Figure 1 (in the relevant range of pair clashes), and each dotted line plots the best results from [3]. The leftmost point on every solid line has the smallest number of clashes possible.

to travel. In 2002-03, for example, if we allow fifteen extra pair clashes, we can save almost 3% on the distance travelled.

Figure 2 shows a subset of the data from Figure 1, but compared with the multi-objective data from [3]. The distinct advantage of the new approach is clear: it produces results that are uniformly better, often by 5–10%.

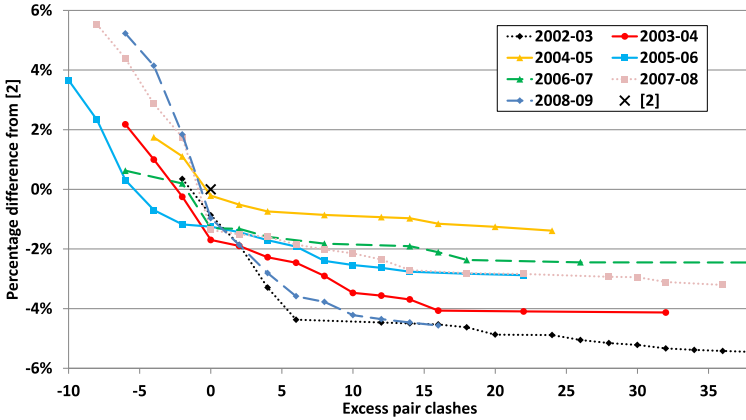


Fig. 3. For the 4-round case, each line plots the best distance result achieved with the corresponding number of pair clashes in excess of the [2] limits, for the same runs as Table 1 plus another set of runs that tried to minimise clashes as far as possible. Negative values indicate reduced distances travelled. The leftmost point on every line has the smallest number of clashes possible, usually sixteen (but fourteen for 2008–09).

Figure 3 shows the complete results on the 4-round fixtures. The overall pattern is the same: as we allow more pair clashes, we reduce the overall distance that has to be travelled. It is noticeable that for the same number of pair clashes (i.e. zero on the graph), the multi-objective approach is able to produce shorter overall travel distances than [2]: occasionally it can also do this for fewer pair clashes (i.e. negative on the graph). Given that the multi-objective approach is also able to offer a set of potential solutions offering varying trade-offs, this is a major advantage of this methodology.

6 Conclusions

We have described a multi-objective evolutionary algorithm that derives schedules for the busy New Year period in the English Football League. The system minimises both away teams’ travel and “pair clashes” where teams which are geographically close play at home on the same day, whilst also managing five constraint objectives which restrict the types of games allowed and the numbers of games allowed in London and in Greater Manchester. The schedules derived are superior to those previously published, especially for larger problem instances: also they offer a range of different trade-offs between the main objectives.

Future work will extend the system to incorporate fairness, as defined in [2]: enforcing a maximum distance for individual teams, additional to the existing objectives. We will also explore enhanced features such as weighted pairings, where clashes between “big teams” are viewed as worse than other clashes.

References

1. Kendall, G.: Scheduling English Football Fixtures over Holiday Periods. *JORS* 59(6), 743–755 (2008)
2. Kendall, G., Westphal, S.: Sports Scheduling: Minimizing Travel for English Football Supporters. In: Uyar, A.S., Ozcan, E., Urquhart, N. (eds.) *Automated Scheduling and Planning*. *SCI*, vol. 505, pp. 61–90. Springer, Heidelberg (2013)
3. Kendall, G., McCollum, B., Cruz, F.R.B., McMullan, P., While, L.: Scheduling English Football Fixtures: Consideration of Two Conflicting Objectives. In: Talbi, E.-G. (ed.) *Hybrid Metaheuristics*. *SCI*, vol. 434, pp. 369–385. Springer, Heidelberg (2013)
4. Duarte, A.R., Ribeiro, C.C.: Referee Assignment in Sports Leagues: Approximate and Exact Multi-objective Approaches. In: *19th International Conference on Multiple Criteria Decision Making*, Auckland, New Zealand, pp. 58–60 (2008)
5. Evans, J.R.: A Microcomputer-based Decision Support System for Scheduling Umpires in the American Baseball League. *Interfaces* 18(6), 42–51 (1988)
6. Barone, L., While, L., Hughes, P., Hingston, P.: Fixture Scheduling for Australian Rules Football using a Multi-objective Evolutionary Algorithm. In: *IEEE CEC*, Vancouver, Canada, pp. 3377–3384 (2006)
7. While, L., Barone, L.: Super 14 Rugby Fixture Scheduling using a Multi-objective Evolutionary Algorithm. In: *IEEE CI-Scheduling*, Honolulu, Hawaii, pp. 35–42 (2007)
8. Craig, S., While, L., Barone, L.: Scheduling for the National Hockey League using a Multi-objective Evolutionary Algorithm. In: Nicholson, A., Li, X. (eds.) *AI 2009*. *LNCS*, vol. 5866, pp. 381–390. Springer, Heidelberg (2009)
9. Kendall, G., Knust, S., Ribeiro, C.C., Urrutia, S.: Scheduling in Sports: an Annotated Bibliography. *Computers & Operations Research* 37(1), 1–19 (2010)
10. Rasmussen, R.V., Trick, M.A.: Round Robin Scheduling — a Survey. *EJOR* 188(3), 617–636 (2008)
11. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: *ICGA*, Urbana-Champaign, Illinois, pp. 416–423 (1993)
12. Coello Coello, C.A., Lamont, G., Van Veldhuizen, D.: *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer (2007)
13. Hingston, P., Barone, L., Huband, S., While, L.: Multi-level Ranking for Constrained Multi-objective Evolutionary Optimisation. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) *PPSN 2006*. *LNCS*, vol. 4193, pp. 563–572. Springer, Heidelberg (2006)
14. Huband, S., Tuppurainen, D., While, L., Barone, L., Hingston, P., Bearman, R.: Maximising Overall Value in Plant Design. *Minerals Eng.* 19(15), 1470–1478 (2006)
15. While, L., Hingston, P.: Usefulness of Infeasible Solutions in Evolutionary Search: an Empirical and Mathematical Study. In: *IEEE CEC*, Mexico, pp. 1363–1370 (2013)

Coupling Evolution and Information Theory for Autonomous Robotic Exploration

Guohua Zhang^{1,2} and Michèle Sebag²

¹ Chengdu Institute of Computer Applications, Chinese Academy of Sciences,
Chengdu 610041, China

² TAO, CNRS – INRIA – LRI, Université Paris-Sud, 91128 Orsay Cedex, France

Abstract. This paper investigates a hybrid two-phase approach toward exploratory behavior in robotics. In a first phase, controllers are evolved to maximize the quantity of information in the sensori-motor datastream generated by the robot. In a second phase, the data acquired by the evolved controllers is used to support an information theory-based controller, selecting the most informative action in each time step.

The approach, referred to as EvITE, is shown to outperform both the evolutionary and the information theory-based approaches standalone, in terms of actual exploration of the arena. Further, the EvITE controller features some generality property, being able to efficiently explore other arenas than the one considered during the first evolutionary phase.

Keywords: Evolutionary robotics, information theory, intrinsic motivation, entropy.

1 Introduction

Quite a few disciplinary fields are concerned with building autonomous robotic controllers, ranging from optimal control to evolutionary robotics (ER) [14], machine learning (ML) and specifically reinforcement learning (RL) [19,6] and artificial intelligence (AI) [16]. For the sake of computational and experimental conveniency, many approaches rely on simulators; the relevance of the resulting controllers thus is subject to the so-called reality gap [10,17].

This paper is concerned with building robotic controllers in an in-situ, simulator-free fashion. Whereas the simulator-free setting sidesteps the reality gap, it raises the challenge of defining intrinsic criteria (optimization objective or decision hints respectively supporting the on-board evolution or decision making process) without any ground truth about the appropriateness of the robot behavior in its environment. This study is primarily motivated by swarm robotics [11], where simulator-based approaches face a super-linear computational complexity w.r.t. the number of robots in the swarm.

The proposed approach takes inspiration from the intrinsic motivation criterion [15,2,12] and from information theory-driven evolutionary robotics [3] (more in section 2), with scalability and generality as main criteria. The presented

approach, referred to as EvITE (*Evolution and Information Theory-based Exploratory Robotics*) and introduced in section 3, combines machine learning and evolutionary principles. Formally, in a first phase, evolutionary controllers are built along the same ideas as in [3]; in a second phase the data acquired by these controllers is used to support an information theory-based controller. The experimental validation of EvITE shows that it outperforms the evolutionary and the IT controllers standalone, in terms of exploration and behavioral diversity [7,9] (section 4). Importantly, EvITE features some *generality* property, in the sense that it also achieves good exploratory performance in other arenas than the one used to train the evolutionary controllers. The paper discusses the complementarity of evolution and ML approaches and how to best combine them to build robotic controllers, and concludes with some perspectives for further research.

2 Related Work

This section discusses a few approaches to autonomous robotics, based on evolutionary robotics (ER) or machine learning (ML) or combining both.

ER (with the exception of [7,9,8], see below) considers a fitness function that independently maps each controller trajectory on \mathbb{R} . A first challenge is to define an appropriate search space, enabling to describe powerful controllers while supporting evolutionary optimization [22]. A second challenge is that ER faces a noisy optimization problem: finding the controller π such that the expectation of the fitness of the trajectories (where the expectation is taken over all trajectories generated from π , reflecting e.g. the robot sensor and motor noise), is maximal. How to deal with this noisy optimization problem, and keep the number of trajectories needed to estimate the expectation within reasonable computational cost has been studied for instance by [5].

In mainstream ML [18], a fixed instant reward on each (state,action) pair is defined under the Markovian assumption that the current reward only depends on the previous state. The learning process then focuses on estimating the value $Q(s, a)$, defined as the maximal cumulative reward the robot can get after selecting action a in state s . An optimal controller immediately follows from the optimal value function, by selecting in state s the action a^* maximizing $Q(s, a)$. (In practice, the learning process alternates between estimating Q and optimizing π ; the details are left out for simplicity, referring the reader to [6] for a comprehensive presentation).

Hybrid ML-ER approaches include Genetic-based Machine Learning, and specifically Learning Classifier Systems variants [20,21], based on the evolutionary learning of sets of (condition-action) or (condition-action-effect) rules. While LCS rules enable in principle to control both the robot and its internal state, GBML seemingly faces scalability issues compared to Neural Nets and even more to new NN-based controller representations [22]. Another hybrid approach, based on the notion of *intrinsic motivation*, is pioneered by [15,2,12]. Let K be a trajectory, sequence of the (state s_t , action a_t) pairs generated by a controller along time, and let \mathcal{K}_i denote the archive of the first i trajectories generated by the

learning/optimization process. A so-called forward model f_i is learned from \mathcal{K}_i , estimating the transition model in the robot environment, that is the next state s' of the robot after selecting action a in state s , where S and A respectively stands for the set of states and set of actions. The key point is that the accuracy of f_i can be estimated on-board during the next trajectory of the robot, as the robot observes the state s_{t+1} yielded by selecting action a_t in state s_t . The accuracy $Acc(f_i)$ of f_i on the next trajectory K thus defines an intrinsic information, accessible to the robot without any external ground truth.

$$f_i : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S} \quad Acc(f_i) = Pr(s_{t+1} = f_i(s_t, a_t) | (s_t, a_t, s_{t+1}) \in K)$$

Note that the above accuracy defines a misleading fitness, as a motionless controller ($s_{t+1} = s_t$) would get a very high fitness. The intrinsic motivation (IM) fitness \mathcal{F}_{IM} therefore associates to a controller the Acc increase:

$$\mathcal{F}_{IM}(\pi) = Acc(f_{i+1}) - Acc(f_i) \quad (1)$$

The optimization of fitness \mathcal{F}_{IM} thus yields controllers which explore new regions of the (state,action) space, providing new samples and thereby ultimately yielding an optimal forward model. Most interestingly, \mathcal{F}_{IM} does not reward the extra-exploration of noisy regions: if a (state,action) region is noisy – due to e.g. stochastic noise in the environment – repeated explorations of this region are useless as they do not improve the forward model accuracy.

A related though simpler approach, based on the so-called curiosity- or discovery-driven fitness, was proposed in [3]. To each trajectory $(s_t, a_t)_{t=1, \dots, T}$ generated by a controller π is associated the entropic fitness \mathcal{F}_E

$$\mathcal{F}_E(\pi) = - \sum_s p_s \log p_s \quad (2)$$

where s ranges over the states visited in the trajectory and p_s is the fraction of time the trajectory visits s . By construction, an optimal controller according to \mathcal{F}_E is one uniformly visiting the state space. Along this line, the entropic fitness would provide good material in order to build an accurate forward model, too. Note however that entropic fitness does not require a forward model to be learned, contrarily to intrinsic motivation. Its limitation compared to IM is that it might tend to over-explore stochastic regions of the state space, if any.

In summary, ML and ER differ in the way they build a controller and use the memory of the building process. The result of the ML process is a value function, and the controller can be explicitly derived from the value function. ER memorizes the evolution process through the ER controller itself, expressed as e.g. the weight vector of a neural net, and through the archive of the past trajectories. This archive makes it feasible to define more sophisticated fitness functions. For instance, [9,8] characterize and exploit the difference between a trajectory and the past ones to enforce the robust and creative sampling of the trajectory space; [7,13] likewise use this diversity, possibly along a multi-objective framework; [3] further defines a discovery-driven fitness, computing the conditional entropy of the current trajectory w.r.t. the trajectory archive.

3 Ev-ITER Overview

This section presents the new EvITE scheme, inspired from [3,12] and hybridizing the evolutionary and the learning approaches in order to build an autonomous exploratory robotic controller. EvITE involves two phases. In the first phase, controllers are evolved as in [3]. In the second phase, the trajectories generated by the evolved controllers are used to initialize an entropic state-action value function. This value function, characterizing the promising state-action pairs, is used to support an information-driven controller, and the entropic value function is accordingly updated. A main issue, scalability-wise, is that the discretization of the state and action space be under the control of the robot, depending on its memory and computational resources.

3.1 Phase 1. Evolutionary Exploration

For the sake of self-containedness, let us remind the formal background of the curiosity-driven evolutionary robotics approach [3]. Let $K = (s_t, a_t)_{t=1}^T$ denote a T -length trajectory, where s_t (respectively a_t) denotes the sensor (resp. motor) value vector at time t ($s_t \in \mathbb{R}^s$, $a_t \in \mathbb{R}^m$). An ϵ -clustering algorithm is used to incrementally cluster the sensor and the motor spaces independently¹ [4]. To each cluster c is associated the fraction p_c of time spent in this cluster (number of times s_t belongs to c , divided by T) and the entropic fitness is defined by Eq. (2).

After a number N of generations, the best controllers in the last population are retained, and the set \mathcal{K} of trajectories they generated is used to initialize the entropic value function.

3.2 Phase 2: Initializing the Entropic Value Function

EvITE starts by discretizing the state and action space. Mainstream clustering algorithms, the k -means and ϵ -clustering algorithms [4] have complementary strengths and weaknesses: while ϵ -clusters are imbalanced, k -means clustering is sensitive to the k hyper-parameter, and might additionally yield unstable clusters. For this reason, we proceed as follows. Let n_s denote the number of ϵ -clusters built from the sensor vectors s_t in the archive \mathcal{K} . Setting the number k of clusters to n_s , P independent runs of k -means clustering algorithm are launched on the set of sensor vectors, and the best clustering after the distortion criterion is retained, where the distortion is measured from the sum of square distance between any vector and the center of the cluster it belongs to. Likewise, letting n_a denote the number of ϵ -clusters built from the motor vectors a_t in the archive \mathcal{K} , n_a clusters are obtained using the best clustering solution obtained out of P independent runs of k -means with $k = n_a$ on the motor vectors in \mathcal{K} .

Let $i(s_t)$ (respectively $j(a_t)$) denote the index of the cluster the sensor vector s_t (resp. the motor vector a_t) belongs to. For the sake of notational simplicity,

¹ An ϵ -clustering algorithm iteratively constructs a set of clusters, by defining a new cluster for each point which is more than ϵ -far from the center of the previous clusters.

it is assumed in the following that \mathcal{K} involves a single trajectory; the robot is said to execute action j in state i when $i(s_t) = i$ and $j(a_t) = j$.

To each pair i, j is associated the list $Z(i, j)$ storing all instants t following the execution of action j in state i (such that $i(s_{t-1}) = i; j(a_{t-1}) = j$). Let $S(i, j)$ denote the multi-set of states for t in $Z(i, j)$, that is the list of all states that the robot visited just after executing action j in state i . Let $Q(i, j)$ denote the entropy of $S(i, j)$: the higher $Q(i, j)$, the less one can predict the behavior of the robot after selecting action j in state i . $Z(i, j)$ and $S(i, j)$ are built and

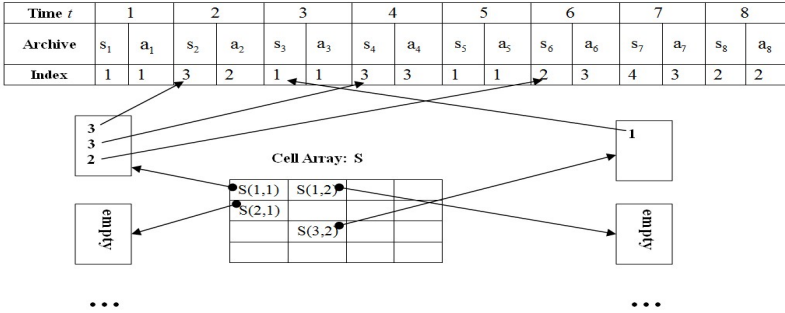


Fig. 1. Computing the entropic value function from a 8-length trajectory (top), with $n_s = n_a = 4$. The 4×4 matrix S is built, where list $S(i, j)$ is used to compute entropy $Q(i, j)$ when not empty.

maintained online (Fig. 1). A sliding window is used to comply with the robot limited memory resources, where only the last λ elements in $Z(i, j)$ and $S(i, j)$ are retained.

3.3 Phase 2. Information-Driven Navigation

Based on the entropic value function, an information-driven controller is defined as follows, considering three modes.

In the first mode, referred to as **babbling mode** and triggered when the robot is visiting a state i where few actions have been tried (the size of $Z(i, j)$ is less than λ , for all actions j), the action cluster index is selected uniformly in $1 \dots n_a$.

The second mode is the main mode, referred to as **curious mode**. In this mode, the action with optimal score is selected, where

$$score(j|i) = (1 - \alpha)Q(i, j) + \alpha(1 - Pr(i|(i, j))) \tag{3}$$

where the last term $Pr(i|(i, j))$ is the probability of staying in state i after selecting action j in state i (estimated from the frequency of $i(s_{t-1}) = i, j(a_{t-1}) = j, i(s_t) = i$ for t in $Z(i, j)$), and α a hyper-parameter controlling the balance between the former and the latter terms. The intuition is that, everything being equal, exploration is better enforced by selecting an action that modifies the robot state.

The third mode, referred to as **exploratory mode**, is meant to prevent the degenerate behaviors possibly incurred in the *curious mode* (e.g. dancing in front of a wall). Here, one simply selects the action less selected the last μ times state i was visited.

In all three modes, letting j denote the index of the selected action cluster at time t , then the actual motor vector a_t is selected uniformly in the j -th action cluster.

Overall, the babbling mode is triggered when arriving in an unknown state. Otherwise, the curious mode is selected with probability 95% and the exploratory mode is selected with probability 5%. In each time step, the $Z(i, j)$ and $S(i, j)$ lists are updated. Formally, after having selected action j in state i at time t , the last elements in $Z(i, j)$ and $S(i, j)$ are removed if necessary (if $|Z(i, j)| > \lambda$) and indices $t + 1$ and $k = i(s_{t+1})$ are respectively added to $Z(i, j)$ and $S(i, j)$.

4 Experimental Validation

This section reports on the experimental validation of EvITE. The primary goal of the experiments is to assess the performance of the proposed scheme in terms of the exploration indicators below, comparatively to the evolutionary curiosity-driven scheme [3] and the intrinsic motivation scheme [12] standalone². Another goal is to assess the generality of the resulting controllers, i.e. their ability to explore new arenas, distinct from the one used in the first phase of EvITE.

For the sake of reproducibility, the experimental setting uses the Webot simulator emulating an E-puck robot with 8 infra-red sensors and 2 motors. The evolutionary curiosity baseline is implemented using same setting as in [3]: the robot population is evolved for 200 generations. The 1st phase in EvITE uses the trajectories of the best robots in the 200-th generation of Curiosity to build the $Z(i, j)$ and $S(i, j)$ data. The intrinsic motivation baseline proceeds using an empty $Z(i, j)$ and $S(i, j)$, based on the fact that $Q(i, j)$ is a proxy for the accuracy of the forward model in state i, j (Eq. 1). The hyper-parameter setting is as follows. Window lengths λ and μ are respectively set to 500 and 60. The ϵ parameter used for respectively clustering the sensor and motor vectors is set to 450 and 3000. Each EvITE and IM run considers a sequence of 2,000 epochs, where an epoch is a robot starting from the same starting point (lower left corner in the arenas, Fig. 2) and navigating in the arena for 2,000 time steps (175 time steps are required to cross the arena at average speed). Three arenas are considered: the easy one is taken from [9,3] (referred to as hard arena in [9], Fig 2.a); a harder one referred to as graph-arena (Fig 2.b); and a maze-like arena (Fig 2.c). The exploratory performance of Curiosity, IM and EvITE is comparatively displayed on the easy (Fig. 2. d), maze (Fig. 2.e) and maze (Fig. 2.f) arenas, showing the number of squares p_1 visited at least once per run (averaged

² The comparison with other ER approaches, specifically [20,22], faces the difficulty of defining a fitness function: building a fitness function from the exploration indicators is inappropriate since computing them requires some ground truth; but the whole motivation of the approach is to handle cases where the ground truth is not available.

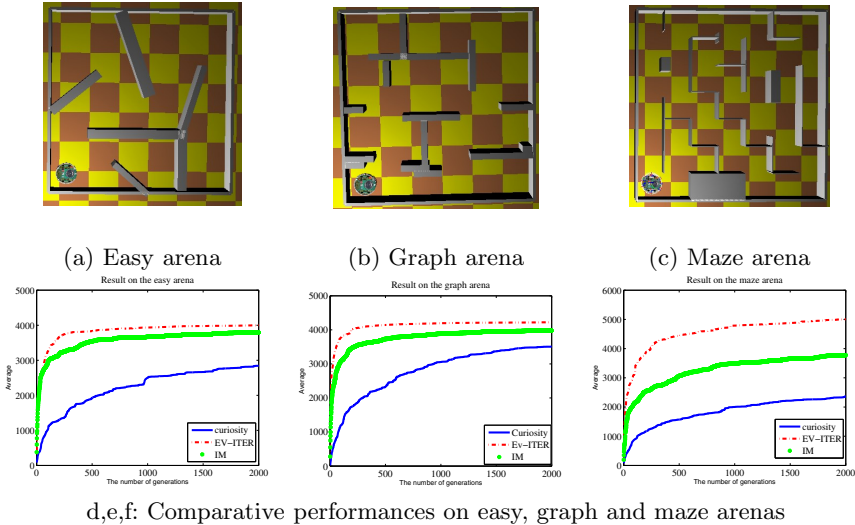


Fig. 2. Comparative performances of Curiosity, IM and EvITE on the (a) easy arena, (b) graph arena and (c) maze arena (respectively $0.6 \text{ m} \times 0.6 \text{ m}$, $0.6 \text{ m} \times 0.6 \text{ m}$, and $0.7 \text{ m} \times 0.7 \text{ m}$). The number of squares visited at least once (each arena involving 10,000 squares for comparison) is reported, averaged out of 15 independent runs. It is reminded that EvITE entropic value function is initialized from the trajectories on the Easy arena, and the curiosity-driven controllers are trained on the Easy arena too.

out of 15 independent runs, 10,000 squares for each arena). Furthermore, this result supports the generality of the EvITE approach, by noting that the EvITE controller performs well on the graph and maze arenas, even though the entropic value function is initialized by training on the easy arena. Comparatively, the curiosity-driven controllers trained on the easy arena do poorly on the other arenas. The performance of EvITE is significantly higher than for the intrinsic motivation alone, which itself outperforms the curiosity-driven approach, all the more so as the complexity of the arena increases. A more detailed account of the performance indicators p_1 and p_2 is reported in Table 1, indicating the average and median number of squares visited once or twice, averaged on 2,000 epochs and 15 runs. The generality is visually assessed on Fig. 3, showing the actual robot trajectories after 500 and 2,000 epochs. The top (resp. middle, bottom) rows display the behavior of Curiosity (resp. EvITE, IM), showing the arena visited during the first 500 epochs and the area visited during all 2,000 epochs, on the easy arena (columns 1 and 2), on the graph arena (3rd column) and on the maze arena (4th column). It is seen that Curiosity is lagging behind the other two approaches in all cases. On the easy and medium arenas, the performances of IM are visually a bit behind those of EvITE for 500 epochs (complementary results omitted due to space restrictions), and they catch up for 2,000 epochs.

Table 1. Comparative performances of EvITE, IM and Curiosity on the easy, graph and maze arenas, reporting the average and median (std.dev.) number of squares visited out of 15 runs

		IM		Curiosity		Ev-ITER	
		1 visits	2 visits	1 visits	2 visits	1 visits	2 visits
Easy arena	Median	3760	1898	2884	2106	4105	2514
	Average (std.dev.)	3796.7 (354.21)	1921.9 (7.5607)	2843.1 (198.9906)	2104 (134.8306)	3999.1 (236.0734)	2501.3 (219.4420)
Graph arena	Median	3693	2123	3232	1960	3995	1774
	Average std.dev.	3974.8 (264.81)	2136.4 (0.11499)	3510.1 (186.6)	1942.1 (24.56)	4222.6 (80.479)	1825.1 (43.036)
Maze arena	Median	3614	1831	2102	1251	4553	2050
	Average std.dev.	3779 (455.31)	1788.2 (155.96)	2362.5 (192.91)	1279.9 (19.21)	5005.5 (245.86)	2082.4 (158.95)

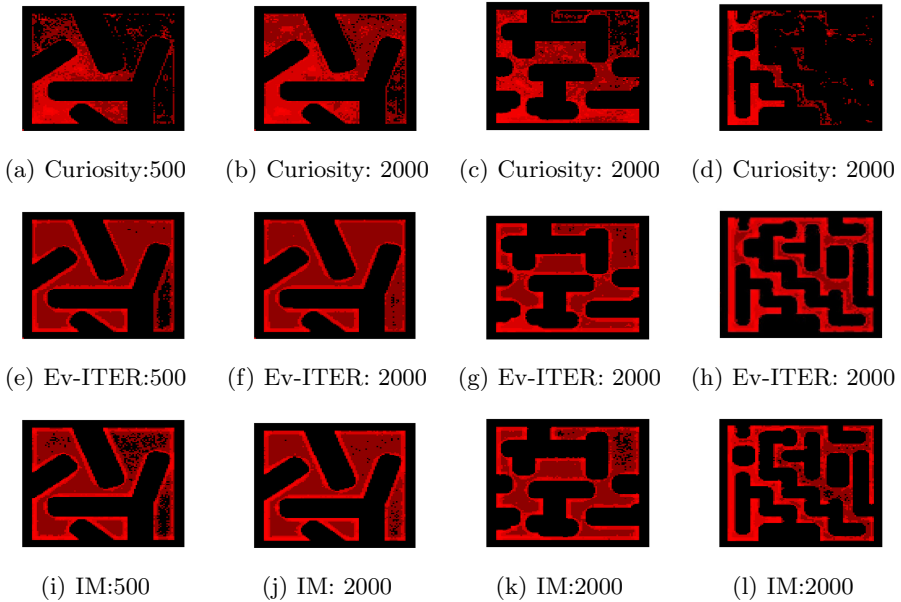


Fig. 3. Trajectories of the evolutionary curiosity (top row), EvITE (middle row) and IM (bottom row) on the easy, graph and maze arenas, cumulative over 500 robots and 2,000 robots

On the maze arena finally, the performances of IM are behind those of EvITE for both 500 and 2,000 epochs (see the middle corridors in the maze).

These results establish the merits of the hybrid EvITE approach. On the one hand, it significantly improves on the evolutionary curiosity approach alone. On the other hand, it also improves on the intrinsic motivation approach, as it is shown to visit more densely the regions far from the starting point. Last but not least, the EvITE controller features a quite decent generality as it reaches good

exploratory performances when the 2nd phase is conducted in another arena than the one considered in the 1st phase. Complementary experiments, omitted due to the space restrictions, show that the exploratory performance is almost as good when the 1st phase is conducted in the easy arena, than when it is conducted in the same arena as the one where the 2nd phase takes place.

5 Discussion and Perspectives

This paper presents a new combination of EC and ML approaches toward autonomous exploration in in-situ robotics. The challenge of defining intrinsic criteria available on-board is addressed by taking inspiration from [3,12]. On the top of these pioneering works, the EvITE scheme proceeds as a 2-phase process. The first phase proceeds exactly as in [3]. The second phase exploits the data gathered in the first phase in order to support a learning approach. The impact of these data is evidenced by comparing the EvITE performances to that of IM. The experimental results (Fig. 2) suggest that the additional information provided to the EvITE controller is never compensated for by the IM controller: the IM performance plateaus well below the EvITE performance.

These results suggest that one strength of the evolutionary approach is to be able to gather relevant data in a fast and efficient manner; once these data have been acquired, then deterministic ML might be in better shape to exploit them thoroughly. The originality of this coupling is to use evolution to provide informative data to ML, whereas the mainstream hybridation scheme uses evolution to optimize the ML solution.

It is worth emphasizing the fact that the EvITE controller can yield good performances also in new arenas, thus featuring some generality property, although admittedly the arenas used in the 2nd phase are not very different from the one used in the 1st phase.

Still the possibility of using different environments in the 1st and 2nd phases opens interesting perspectives, particularly in terms of *safe* robotic training. For the sake of a safe in-situ robot training, it makes sense indeed to consider a simple and dangerless arena to acquire the initial data, and to launch the robot in a new and possibly more hazardous arena, using the data as prior knowledge to prevent or mitigate the dangers of blind exploration. Another perspective for further study is to incrementally revise and specialise the sensor and action clusters considered by EvITE, along the same lines as in [12]. A third perspective is to involve the user in the loop along an interactive optimization setting [1], with the goal of achieving other target behaviors on the top of the exploratory behavior.

References

1. Akrou, R., Schoenauer, M., Sebag, M.: Preference-based policy learning. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part I. LNCS, vol. 6911, pp. 12–27. Springer, Heidelberg (2011)

2. Baranès, A., Oudeyer, P.Y.: R-iac: Robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development* 1(3), 155–169 (2009)
3. Delarboulas, P., Schoenauer, M., Sebag, M.: Open-ended evolutionary robotics: an information theoretic approach. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *PPSN XI. LNCS*, vol. 6238, pp. 334–343. Springer, Heidelberg (2010)
4. Duda, P.O., Hart, P.E.: *Pattern Classification and Scene analysis*. John Wiley and sons (1973)
5. Heidrich-Meisner, V., Igel, C.: Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In: *Int. Conf. on Machine Learning*, pp. 401–408 (2009)
6. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. *The Int. J. of Robotics Research* 32(11), 1238–1274 (2013)
7. Koos, S., Mouret, J.B., Doncieux, S.: The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Trans. on Evolutionary Computation* 17(1), 122–145 (2013)
8. Lehman, J., Risi, S., D'Ambrosio, D.B., Stanley, K.O.: Rewarding reactivity to evolve robust controllers without multiple trials or noise. *Artificial Life* 13, 379–386 (2012)
9. Lehman, J., Stanley, K.O.: Exploiting open-endedness to solve problems through the search for novelty. *Artificial Life* 11, 329 (2008)
10. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. *Nature* 406(6799), 974–978 (2000)
11. Liu, W., Winfield, A.F.: Modeling and optimization of adaptive foraging in swarm robotic systems. *The Int. J. of Robotics Research* 29(14), 1743–1760 (2010)
12. Lopes, M., Lang, T., Toussaint, M., Oudeyer, P.-Y.: Exploration in Model-based Reinforcement Learning by Empirically Estimating Learning Progress. In: *NIPS*, pp. 206–214 (2012)
13. Mouret, J.B., Doncieux, S.: Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary Computation* 20(1), 91–133 (2012)
14. Nolfi, S., Floreano, D., Floreano, D.: *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT Press, Cambridge (2000)
15. Oudeyer, P.Y., Kaplan, F., Hafner, V.V.: Intrinsic motivation systems for autonomous mental development. *IEEE Trans. on Evolutionary Computation* 11(2), 265–286 (2007)
16. Pfeifer, R., Gomez, G.: Interacting with the real world: design principles for intelligent systems. *Artificial life and Robotics* 9(1), 1–6 (2005)
17. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. *The Int. J. of Robotics Research* 27(2), 157–173 (2008)
18. Sutton, R., Barto, A.: *Reinforcement learning: An introduction*. Cambridge Univ. Press (1998)
19. Thrun, S., Burgard, W., Fox, D.: *Probabilistic Robotics*. MIT Press (2005)
20. Williams, H., Browne, W.N.: Integration of Learning Classifier Systems with simultaneous localisation and mapping for autonomous robotics. In: *CEC 2012*, pp. 1–8 (2012)
21. Hurst, J., Bull, L., Melhuish, C.: TCS learning classifier system controller on a real robot. 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. 588–597. Springer, Heidelberg (2002)
22. Koutnk, J., Cuccu, G., Schmidhuber, J.: Evolving large-scale neural networks for vision-based reinforcement learning. In: *GECCO 2013*, pp. 1061–1068 (2013)

Local Optima and Weight Distribution in the Number Partitioning Problem

Khulood Alyahya and Jonathan E. Rowe

School of Computer Science
University of Birmingham, B15 2TT, UK
{kya020, J.E.Rowe}@cs.bham.ac.uk
<http://www.cs.bham.ac.uk>

Abstract. This paper investigates the relation between the distribution of the weights and the number of local optima in the Number Partitioning Problem (NPP). The number of local optima in the 1-bit flip landscape was found to be strongly and negatively correlated with the coefficient of variation (*CV*) of the weights. The average local search cost using the 1-bit flip operator was also found to be strongly and negatively correlated with the *CV* of the weights. A formula based on the *CV* of the weights for estimating the average number of local optima in the 1-bit flip landscape is proposed in the paper. The paper also shows that the *CV* of the weights has a potentially useful application in guiding the choice of heuristic search algorithm.

Keywords: Combinatorial optimisation, phase transition, partitioning problem, makespan scheduling, fitness landscape.

1 Introduction

The number partitioning problem (NPP) is a classical problem in theoretical computer science. It is one of Garey and Johnson's six basic NP-complete problems [4]. It has many practical applications such as multiprocessor scheduling. The optimisation version of the problem can be defined as: given a set $A = \{a_1, \dots, a_n\}$ of positive integers (weights) drawn at random from the set $\{1, 2, \dots, M\}$, the goal is to partition A into two disjoint subsets S, S' such that the discrepancy between them $|\sum_{a_i \in S} a_i - \sum_{a_i \in S'} a_i|$ is minimised. A partition is called perfect, if the discrepancy between the two subsets is 0 when the sum of the original set is even, or 1 when the sum is odd. Equivalently, the problem can be viewed as minimising the maximum sum over the two subsets:

$$f(x) = \max \left\{ \sum_{a_i \in A} a_i x_i, \sum_{a_i \in A} a_i (1 - x_i) \right\}, x \in \{0, 1\}^n \quad (1)$$

The NPP has an easy-hard phase transition, with many perfect partitions with probability tending to 1 in the easy phase, then the number of perfect partitions drops to zero with probability tending to 1 in the hard phase. The transition is

determined by the control parameter $k = \log_2 M/n$, which corresponds to the number of bits required to encode the numbers in the set divided by the size of the set. For $\log_2 M$ and n tending to infinity, the transition occurs at the critical value of $k_c = 1$ [2,6]. The pairs of weights that are placed in the same subset or in opposite subsets in all optimal solutions of an NPP instance, form the backbone of that instance. There is a very sharp increase in the backbone size of the optimal solutions in the NPP as one approaches the phase transition boundary, after which the backbone tends to be complete giving a unique optimal solution [8].

Theoretical analysis of randomized local search shows that it can be a good approximation algorithm for the NPP [12]. Generally, when using local search metaheuristics the average local search cost can vary across problem instances of the same size by many orders of magnitude. A number of models of local search cost for various NP-complete problems such as the Traveling Salesman Problem (TSP), the Boolean Satisfiability Problem (SAT), and the Job-Shop Scheduling Problem (JSP) [7,13,11], have been developed as functions of the problem fitness landscape features. These models attempt to provide explanations for some or ideally all of this variability in the search cost. They also aim to give insights into why one problem instance is more difficult than the other and how the local search is influenced by the properties of the landscape.

The phase transition in the NPP provides an explanation for the increase in the local search cost for problem instances in the hard phase [1], agreeing with the intuition that having fewer optimal solutions usually yield an increase in the local search cost. However, there is considerable variability in the average local search cost for instances in the hard phase that the phase transition fails to account for. The average local search cost was found to vary by many orders of magnitude for hard instances drawn from different distributions such as uniform, normal, negatively and positively skewed distributions [1]. Also, it has been shown in [9] that the number of local optima is not dependent on the easy-hard phase transition, which suggests that the control parameter k is not necessarily good for predicting which problem instances will be easy or hard for local search.

In this paper, we investigate the relation between the distributions of the weights and the fitness landscape features of the NPP. We examine how the number of local optima and the average cost of local search are influenced by the distributions of the weights.

2 Weights Coefficient of Variation and NPP Landscape

Most of the existing analyses of the NPP assume that weights are drawn uniformly at random from a given range. Paper [1] shows that when different distributions are used, there can be large changes in local search performance in hard instances, most noticeable in the 1-bit flip landscape. These changes are mostly due to the difference in the number of local optima between instances drawn from the different distributions. The results shown in [1] suggest that the variability of the weights is what causes the difference in the number of local

optima between instances of the NPP. To measure the variability of the weights we use the coefficient of variation (*CV*) which provides a measure of relative variability or dispersion. *CV* is defined as the ratio of the standard deviation σ to the mean μ :

$$CV = \frac{\sigma}{\mu} \quad (2)$$

We conducted a series of experiments in order to test the assumption that variability of the weights results in different number of local optima. In the experiments, instances from small problem sizes were considered to allow exhaustive enumeration of the entire search space, the problem sizes considered are $n = 12, 14, 16, 18, 20$. For each problem size, 700 instances from the hard phase ($k = \log_2 M/n > 1$) were randomly generated with different values of *CV*.

The rest of this section gives an analysis of the obtained experimental results, focusing on the number of local optima and the average local search cost and how they correlate with the *CV*. The following definitions will be used throughout this paper:

Search Space. The search space X is the finite set of all the candidate solutions. Since the fitness function of NPP is a pseudo-Boolean function the search space size is 2^n . The binary representation of NPP creates a symmetry in the search space, in the sense that a solution and its bitwise complement have the same fitness value. Thus, the number of unique solutions is 2^{n-1} .

Neighbourhood. A neighbourhood is a mapping $N : X \rightarrow P(X)$, that associates each solution with a set of candidate solutions, called neighbours which can be reached by applying the neighbourhood operator once. The set of neighbours of x is called $N(x)$, and $x \notin N(x)$.

We consider two different neighbourhood operators in this paper: The 1 hamming operator (H1), the neighbourhood using this operator is the set of points that are reached by 1-bit flip mutation of the current solution x , hence the neighbourhood size is $|N(x)| = n$. The second operator is the 1+2 hamming operator (H1+2), the neighbourhood here includes the hamming one neighbours plus the hamming two neighbours of the current solution x which can be reached by 2-bits flip mutation, the neighbourhood size for this operator is $|N(x)| = n + (n(n-1)/2)$.

Fitness Landscape. The fitness landscape of a combinatorial optimisation problem is a triple (X, N, f) , where f is the objective function $f : X \rightarrow R$, X is the search space and N is the neighbourhood operator function [10].

Local Optima. A point $x \in X$ is a local optimum iff $\forall y \in N(x), f(y) \geq f(x)$. The number of local optima found in the fitness landscape will be referred to as m .

2.1 Number of Local Optima

We investigate here if the variability of the weights correlates with the number of local optima in the NPP landscapes induced by the H1 and H1+2 neighbourhood operators. Figure 1 shows that the local optima fraction of the search

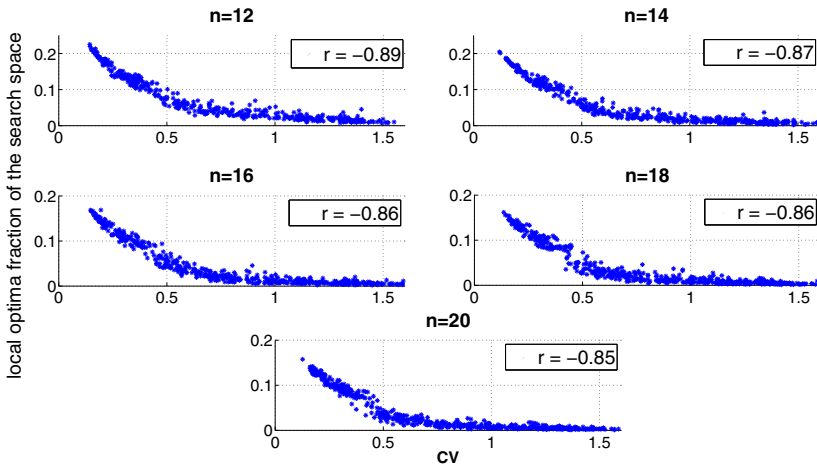


Fig. 1. Local optima as a fraction of the search space in the H1 landscape versus the coefficient of variation CV for problem sizes $n = 12, 14, 16, 18$ and 20 and for 700 instances for each problem size. The correlation coefficients (r) between CV and the the fraction of the local optima are shown for each plot.

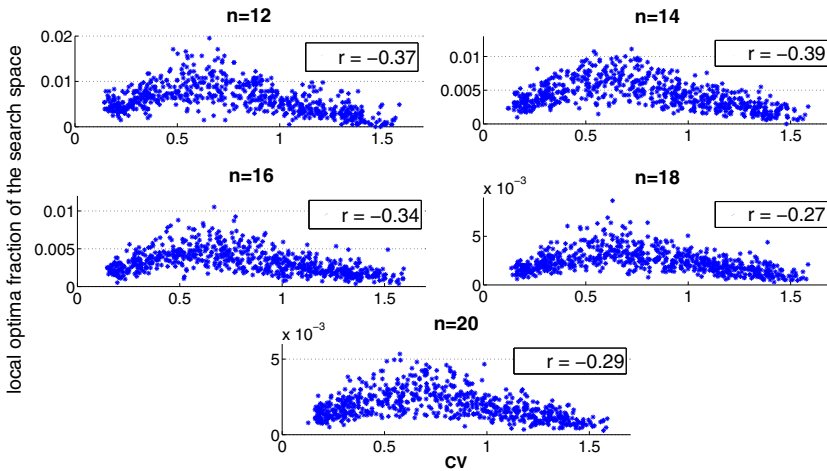


Fig. 2. Local optima as a fraction of the search space in the H1+2 landscape versus the coefficient of variation CV for problem sizes $n = 12, 14, 16, 18$ and 20 and for 700 instances for each problem size. The correlation coefficients (r) between CV and the the fraction of the local optima are shown for each plot.

space ($m/2^n$) in the H1 landscape is highly and negatively correlated with the coefficient of variation.

The intuition behind the strong correlation between the number of local optima in the H1 landscape and the CV is that, for smaller values of CV , the similarity of the weights provides many ways to arrange the weights such that moving one of the weights from one subset to the other does not lead to a better solution, resulting in a larger number of local optima in the H1 Landscape. Contrarily, in instances with larger values of CV , the discrepancy of the weights enables the same application of the 1-bit flip move operator to lead to a better solution most of the time, which result in fewer local optima.

Figure 2 shows that the fraction of local optima in the H1+2 landscape has a weak negative correlation with the CV and slightly higher fractions of local optima for instances with ($0.4 < CV < 0.8$). In both landscapes, the fraction of local optima decreases as n gets larger. The correlation coefficients also get weaker as the problem size grows, with a faster decay in the correlation between the CV and the number of local optima in the H1+2 landscape. Higher orders of n would need to be studied to examine if and how the correlation between the CV and the number of local optima in both landscapes continue to exist in larger problem sizes.

2.2 Average Number of Local Optima

The number of local optima typically influences the performance of local search metaheuristics, and for the NPP it has been shown in [1] that the number of local optima does indeed influence the local search cost. Given that, it is interesting to be able to estimate the average number of local optima in the landscape of a given NPP instance. For instances of the NPP with weights drawn from a uniform distribution, Ferreira and Fontanari [3] derived the following formula, using statistical mechanics analysis, for the average fraction of local minima in the H1 landscape.

$$\frac{\langle m \rangle}{2^n} = \sqrt{\frac{24}{\pi}} n^{-3/2} \quad (3)$$

Here we propose a generalized formula for estimating the average fraction of local minima in the H1 landscape of the NPP. The formula does not require the knowledge of the distribution from which the weights are drawn and only depends on the CV of the weights and the size of the problem. The proposed formula is based on the data we observed in figure 1 and it is as follows:

$$\frac{\langle m \rangle}{2^n} = a \exp(-b CV) \quad (4)$$

Where the values of the coefficients a and b depend on the size of the problem. Figure 3 shows the estimation of the fraction of the local optima using this formula and with the values of a and b , determined by the least squares fit method, as shown in figure 4.

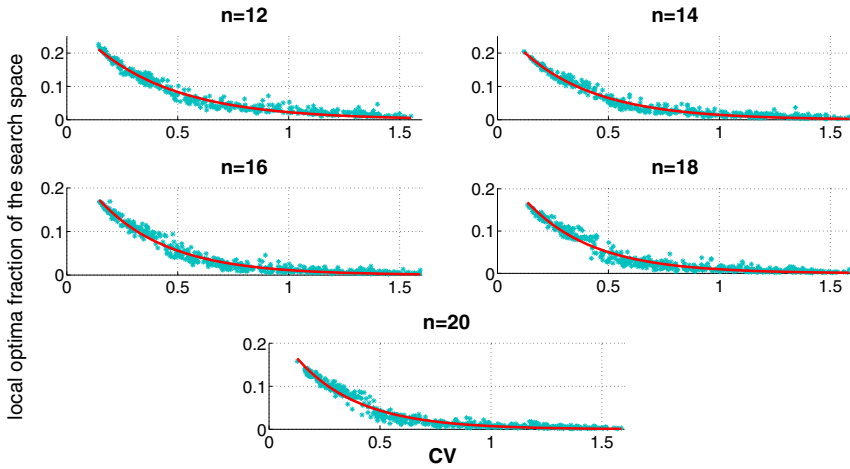


Fig. 3. The fraction of Local optima versus the coefficient of variation CV for problem sizes $n = 12, 14, 16, 18$ and 20 and for 700 instances for each problem size. The least-squares fit lines were obtained using Eq. (4) with values of a and b as shown in figure 4 and the r^2 values for the regression models ≈ 0.97 .

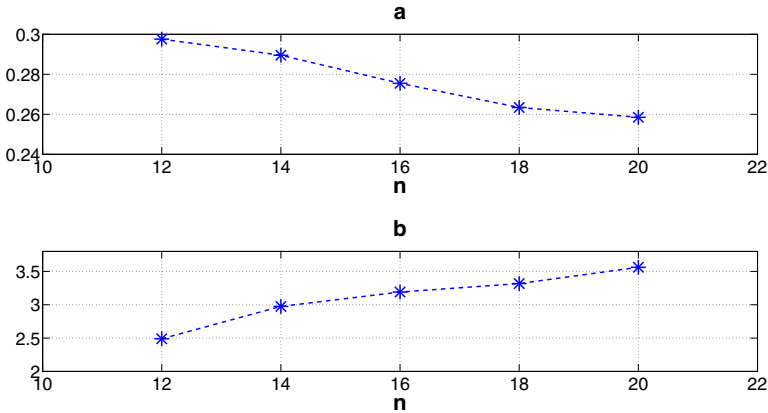


Fig. 4. The values of the a and b coefficients from Eq. (4) for problem sizes $n = 12, 14, 16, 18$ and 20 , estimated using regression models. The r^2 values for the regression models are 0.96, 0.97, 0.97, 0.96, 0.97 respectively.

2.3 Cost of Local Search

To examine how the cost of finding the optimal solution varies for different values of CV and to investigate if the coefficient of variation can be used to guide the choice of local search neighbourhood operator, steepest descent with random

restart (Algorithm 1) was run with two neighbourhood operators, the H1 operator and the larger neighbourhood operator H1+2. The algorithm was run for 1000 times for each instance. The cost of finding the global optima is then calculated using the number of used fitness evaluations.

Algorithm 1. Steepest Descent with Random Restarts

```

repeat
  Chose  $x \in \{0, 1\}^n$ , uniformly at random
  repeat
    choose  $x' \in N(x)$ , such that  $f(x') = \min_{y \in N(x)} f(y)$ 
    replace  $x$  with  $x'$  if  $f(x') < f(x)$ 
  until  $f(x) \leq f(x')$ 
until  $f(x)$  is the optimal solution
  
```

Figures 5 and 6 show, respectively, the results of the steepest descent runs with H1 and H1+2 neighbourhoods operators. The figures show that the average cost of local search using H1 operator and the CV of the weights are strongly and negatively correlated, while for the H1+2 the correlation is weakly negative.

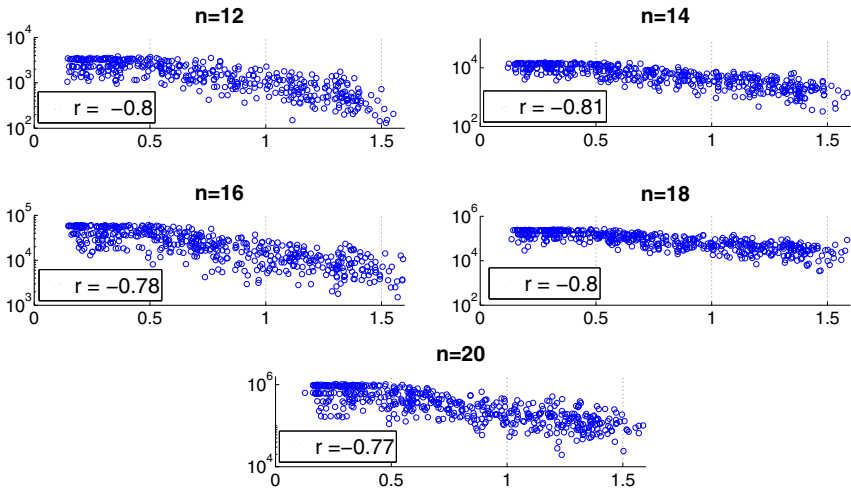


Fig. 5. The cost of steepest descent search with H1 neighbourhood operator for problem sizes $n = 12, 14, 16, 18$ and 20 and for the 700 instances for each problem size. The x-axes represent the coefficient of variation CV and the y-axes represent the average number of fitness evaluations used to find the global optimum in log scale. Each data point represents the average of 1000 runs of steepest descent. The correlation coefficients (r) between CV and the cost of local search are shown for each plot.

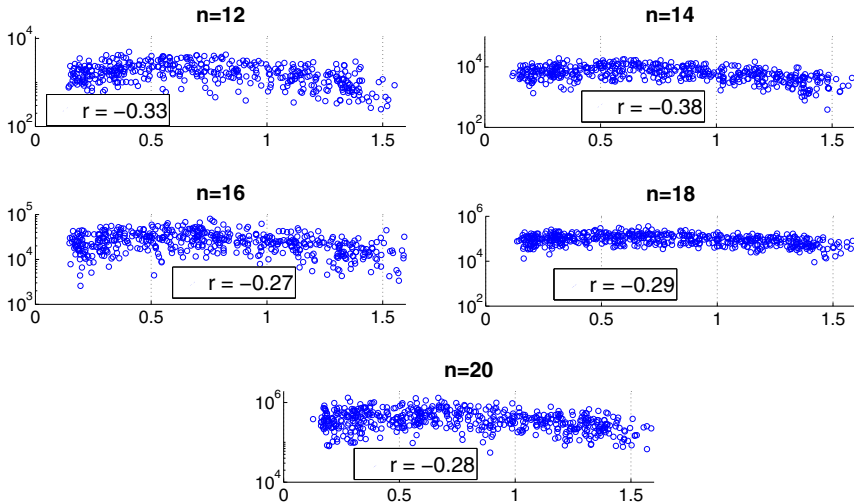


Fig. 6. The cost of steepest descent search with H1+2 neighbourhood operator for problem sizes $n = 12, 14, 16, 18$ and 20 and for the 700 instances for each problem size. The x-axes represent the coefficient of variation CV and the y-axes represent the average number of fitness evaluations used to find the global optimum in log scale. Each data point represents the average of 1000 runs of steepest descent. The correlation coefficients (r) between CV and the cost of local search are shown for each plot.

The landscape induced by the H1+2 operator has far less number of local optima than the landscape induced by the H1 operator and the difference between the number of local optima between the two landscapes is very large for smaller values of CV . Intuitively, a decrease in the number of local optima should yield a decrease in local search cost but if this decrease is not large (i.e. if the difference between the number of local optima between the two landscapes is not large) then it is possible that the advantage of having less local optima be offset by the number of fitness evaluations needed to explore the much larger neighbourhood of the H1+2 operator. To identify the values of CV , where such decrease in the number of local optima would make the use of H1+2 neighbourhood operator be more effective than the H1 neighbourhood operator, we compared the performance of the two operators. Figure 7 shows the number of instances where the performance of the algorithm with the H1 operator was significantly better than the H1+2 performance, the number of instances where the performance of the algorithm with the H1+2 operator was significantly better than the H1 performance, and the instances where the two performances were not statistically significantly different. The Wilcoxon rank-sum test was used to determine the significance of the differences between the performances of the algorithm ($p < 0.05$).

The figure shows that instances with small CV values ($CV < 0.5$), the performance of the H1+2 operator is better than the H1 operator, which is not

surprising due to the low number of local optima in the H1+2 landscape and the very big difference between the number of local optima in the H1+2 landscape compared to the H1 landscape which suggest that the algorithm probably had to do far less restarts when using the H1+2 operator. For instances with large values of CV ($CV \geq 0.5$), the performance of the H1 operator is better than H1+2 operator, even though the number of local optima is less in the landscapes induced by the H1+2 operator. This could be explained by the large number of fitness evaluations needed to explore the much larger neighbourhood of the H1+2 operator. These results show that the CV of the weights has a potentially useful application in guiding the selection of the most suitable neighbourhood operator for a given NPP instance.

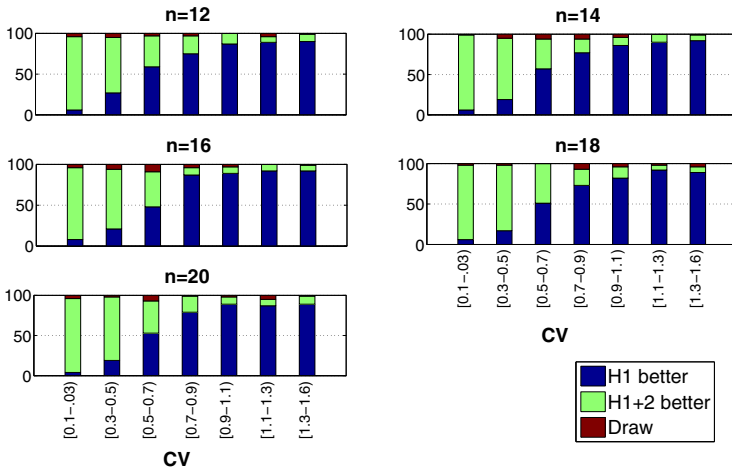


Fig. 7. Coefficient of variation CV against number of instances steepest descent with H1 neighbourhood preformed significantly better, number of instances steepest descent with H1+2 neighbourhood preformed significantly better, and instances where the performance of the two neighbourhood operators was not significantly different. For each problem instance steepest descent was run for 1000 times. The Wilcoxon rank-sum test was used to determine the significance of the differences.

3 Conclusions

In this paper, we examined how the variability of the weights influences the NPP landscape by looking at how the landscape features of the NPP change with the different values of the coefficient of variation (CV) of the weights. The CV of the weights can be easily calculated for a given instance of NPP, and does not require the knowledge of the distribution from which the weights are drawn. We found that the number of local optima and the average cost of local search in the H1 landscape are strongly and negatively correlated with the CV . For the landscapes induced by the H1+2 operator, we found that both the number of

local optima and the average search cost have a weak negative correlation with the CV . We also showed what could be a practical use of the CV of the weights for guiding the choice of the move operators of local search heuristics.

We proposed a formula to estimate the average number of local optima in the H1 landscape that depends only on the problem size and the CV of the weights, exploiting the strong correlation between the CV and the number of local optima in the H1 landscape. We still need as future work to look at larger problem sizes to be able to predict the relation between the two coefficients (a and b) in the formula and the size of the problem. For that we are going to use some sampling techniques to estimate the number of local optima [5].

References

1. Alyahya, K., Rowe, J.E.: Phase transition and landscape properties of the number partitioning problem. In: Blum, C., Ochoa, G. (eds.) *EvoCOP 2014*. LNCS, vol. 8600, pp. 206–217. Springer, Heidelberg (2014)
2. Borgs, C., Chayes, J., Pittel, B.: Phase transition and finite-size scaling for the integer partitioning problem. *Random Structures & Algorithms* 19(3-4), 247–288 (2001)
3. Ferreira, F.F., Fontanari, J.F.: Probabilistic analysis of the number partitioning problem. *Journal of Physics A: Mathematical and General* 31(15), 3417 (1998)
4. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of books in the mathematical sciences. W. H. Freeman (1979)
5. Garnier, J., Kallel, L.: How to detect all maxima of a function. In: *Theoretical Aspects of Evolutionary Computing*. Natural Computing Series, pp. 343–370. Springer, Heidelberg (2001)
6. Mertens, S.: A physicist's approach to number partitioning. *Theoretical Computer Science* 265(1-2), 79–108 (2001)
7. Prügel-Bennett, A., Tayarani-Najaran, M.: Maximum satisfiability: Anatomy of the fitness landscape for a hard combinatorial optimization problem. *IEEE Transactions on Evolutionary Computation* 16(3), 319 (2012)
8. Slaney, J., Walsh, T.: Backbones in optimization and approximation. In: *IJCAI*, pp. 254–259 (2001)
9. Stadler, P.F., Hordijk, W., Fontanari, J.F.: Phase transition and landscape statistics of the number partitioning problem. *Physical Review E* 67(5), 056701 (2003)
10. Stadler, P.F., Stephens, C.R.: Landscapes and effective fitness. *Comments on Theoretical Biology* 8(4-5), 389–431 (2002)
11. Watson, J.P., Whitley, L.D., Howe, A.E.: Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search. *J. Artif. Int. Res.* 24(1), 221–261 (2005)
12. Witt, C.: Worst-case and average-case approximations by simple randomized search heuristics. In: Diekert, V., Durand, B. (eds.) *STACS 2005*. LNCS, vol. 3404, pp. 44–56. Springer, Heidelberg (2005)
13. Zhang, W.: Configuration landscape analysis and backbone guided local search.: Part i: Satisfiability and maximum satisfiability. *Artificial Intelligence* 158(1), 1–26 (2004)

Quasi-Stability of Real Coded Finite Populations

Jarosław Arabas and Rafał Biedrzycki

Institute of Electronic Systems, Warsaw University of Technology, Poland
{jarabas,rbiedrzy}@elka.pw.edu.pl

Abstract. This contribution analyzes dynamics of mean and variance of real chromosomes in consecutive populations of an Evolutionary Algorithm with selection and mutation. Quasi-stable state is characterized with an area in which population mean and variance will remain roughly unchanged for many generations. Size of the area can be indirectly estimated from the infinite population analysis and is influenced by the population size, selection type and parameter, and the mutation variance. The paper gives formulas that define this influence and illustrates them with numerical examples.

Keywords: Population diversity, response to selection, quasi-stability.

1 Introduction

Quasi-stability of populations is a state of an Evolutionary Algorithm (EA) when populations fluctuate in the same area of the search space for many consecutive generations. In effect, although the actual mean and variance of position of chromosomes are variable, they fluctuate around characteristic values which do not change. Expected values of population mean and variance have been analytically derived in [1] assuming that the fitness function is Gaussian (for fitness proportionate selection) or odd and concave (for other selection schemes).

Dynamics of the EA is usually put in the context of convergence or expected hitting time. Here another perspective is taken, where more attention is paid to the population diversity. Several authors have already taken this perspective before in the area of real coded EAs. For example, Beyer and Deb [2] analyzed the dynamics of the population mean and variance assuming a “flat fitness” model (identical selection probabilities) and various crossover schemes, without mutation. They identified a risk for an EA with identical selection probabilities that the population variance may collapse or blow up if the crossover parameters are improperly set and they came up with admissible levels of these parameters. They have also performed similar analysis for the Evolution Strategy for the flat fitness. Works of Arnold have gone in a similar direction. He considered the dynamics of central moments of the population distribution for the Evolution Strategy, assuming a random fitness and mixtures of a deterministic fitness function and noise [3,4].

An important role in the analysis of EA dynamics in real spaces plays the infinite population model introduced by Qi and Palmieri [5]. They considered the

dynamics of populations assuming that they are infinite. This allowed to apply deterministic equations to transform probability density functions of consecutive populations. Their results have been adopted by Karcz-Duleba to analyze the dynamics of populations for functions with a single optimum and with two local optima [6].

In this paper we use results of the population diversity analysis based on infinite population model which have been published in [1] for various types of selection, with and without crossover, with and without elitism. We show that the quasi-stable behavior of an finite population EA can be well explained by this model.

2 Subject of Analysis

We consider a simple EA (see Fig. 1) which combines selection and mutation to maximize a fitness function $q : R \rightarrow R$. The algorithm in the t -th generation is characterized by the base population P^t which contains μ individuals. Each individual P_i^t is a real number. In each generation, a population R^t of reproduced individuals is created by selecting with replacement individuals from the population P^t . Each reproduced individual R_i^t undergoes Gaussian mutation with variance v_m and yields an offspring which becomes the i -th member of the base population for the next generation. The algorithm is stopped after reaching a specified number of generations T .

```

initialize( $P^0$ )
evaluate( $P^0$ )
for  $t = 1$  to  $T$  do
  for  $i = 1$  to  $\mu$  do
     $R_i^t \leftarrow P_j^t$  where  $j \leftarrow \text{select}(P^t)$ 
     $P_i^{t+1} \leftarrow R_i^t + M_i^t$  where  $M_i^t \sim N(0, v_m)$ 
  end for
  evaluate( $P^{t+1}$ )
end for

```

Fig. 1. Outline of the EA under consideration

Empirical moments of the population P^t , the mean \bar{P}^t and the sample variance $s^2(P^t)$, are defined as

$$\bar{P}^t = \frac{1}{\mu} \sum_{i=1}^{\mu} P_i^t \quad s^2(P^t) = \frac{1}{\mu - 1} \sum_{i=1}^{\mu} (P_i^t - \bar{P}^t)^2 \tag{1}$$

Population P^t can be characterized by its state $u_t = [\bar{P}^t, s^2(P^t)]$ which is defined as a pair of mean and sample variance of chromosomes contained by P^t . Thus a single EA run generates a trajectory of states u_t .

The infinite population models have been used to analyze expected values of empirical moments of populations [1,6]. Infinite population size facilitates the analysis since the transformation state u_t to u_{t+1} is deterministic according to equations

$$m_P^{t+1} = m_R^t, \quad v_P^{t+1} = v_R^t + v_m \quad (2)$$

where m_R^t and v_R^t are the expected value and variance of reproduced chromosomes

$$m_R^t = \int_{-\infty}^{\infty} x f_R^t(x) dx, \quad v_R^t = \int_{-\infty}^{\infty} (x - m_R^t)^2 f_R^t(x) dx \quad (3)$$

and f_R^t is the p.d.f. of reproduced points provided that P^t is infinite and distributed with expectation m_P^t and variance v_P^t . If the fitness function is unimodal and even then iteration of equations (2) yields stable values m_P and v_P which have been derived in [1] for various types of selection, with and without crossover, with and without elitism. In general, distribution of f_P^t may be different than normal. Yet, as it has been shown in [1], values of m_P and v_P can be analytically derived with a small error assuming normality of f_P^t .

3 Finite Populations Generated with Stable Expectation and Variance

For a considered EA, populations will never settle down in any position, since state u_{t+1} relates to u_t in a stochastic fashion. Nevertheless, it may be usually observed that populations will fluctuate for many generations in a certain area. In effect, mean values of the population mean and variance in consecutive populations will agree with values predicted from the infinite population model, as it has been shown in [1]. On the other hand, if the population size is finite and variance of chromosomes is bounded, then stabilization of populations will be observed for functions that are not necessarily unimodal — it is only necessary that fitness function is unimodal in sufficiently large neighborhood of local maximum which is covered by all possible locations of populations. Size of this neighborhood can be predicted by analyzing the distribution of mean and variance of finite size populations which are generated by the EA, whose stable expectation and variance have been predicted with the infinite population model.

The population P^t is modeled as a vector of variates of random variables P_1, \dots, P_μ which generate chromosomes P_1^t, \dots, P_μ^t . Since each chromosome is generated according to the same procedure, variables P_1, \dots, P_μ are identically distributed. A stable state $u_P = [m_P, v_P]$ of an infinite population model is assumed. This means that all populations are approximated as if they were generated with the normal distribution with expectation m_P and variance v_P .

Empirical moments of P^t , defined by equation (1), are estimators of theoretical moments. Since the contents of population is random, empirical moments themselves are random variables. It holds

$$E[\bar{P}^t] = m_P, \quad E[s^2(P^t)] = v_P \quad (4)$$

where $E[\cdot]$ denotes the expected value. Further on, it is assumed that $m_P = 0$ to simplify notation without generality loss. If the chromosomes were independent and normally distributed then the population mean would be normally distributed with variance $V[\bar{P}^t] = m_P/\mu$ and the population variance would be chi-square distributed with variance $V[s^2(P^t)] = 2v_P/(\mu - 1)$.

Chromosomes contained by the population P^t result from selection with replacement and mutation of chromosomes from P^{t-1} . For this reason, it is possible that a pair of chromosomes $P_i^t, P_j^t \in P^t$ was generated by mutation of the same parent reproduced from the population P^{t-1} . Therefore random variables P_1, \dots, P_μ which generate populations *are not independent*. Variance of the population mean is then defined as

$$\begin{aligned} V[\bar{P}^t] &= E \left[\frac{1}{\mu^2} \sum_{i=1}^{\mu} \sum_{j=1}^{\mu} P_i^t P_j^t \right] = E \left[\frac{1}{\mu^2} \sum_{i=1}^{\mu} (P_i^t)^2 + \frac{2}{\mu^2} \sum_{i=1}^{\mu-1} \sum_{j=i}^{\mu} P_i^t P_j^t \right] \\ &= \frac{1}{\mu} E[(P^t)^2] + \frac{\mu-1}{\mu} E[P_i^t P_j^t | i \neq j] = \frac{1}{\mu} v_P + \frac{\mu-1}{\mu} r_P^t \end{aligned} \tag{5}$$

where $E[\cdot | \cdot]$ is the conditional expected value and r_P^t is the covariance coefficient between any pair of variables P_i^t, P_j^t .

In a pair of populations P^t and P^{t+1} , any point P_i^{t+1} will result from mutation of a reproduced point, hence $P_i^{t+1} = R_k^t + M_i^t$, where M_i^t is the i -th mutation in generation t which is independent of all other mutations; therefore

$$r_P^{t+1} = E [(R_i^t + M_i^t)(R_j^t + M_j^t)] = E [R_i^t R_j^t] \tag{6}$$

To derive value of r_P^{t+1} consider a pair of distinct chromosomes P_i^{t+1} and P_j^{t+1} . Observe that two scenarios are possible. In the first scenario, which holds with probability a , both chromosomes will be mutants of the same reproduced chromosomes R_k^t . In the second scenario, their parents will be distinct:

$$r_P^{t+1} = aE [(R^t)^2] + (1 - a)E [R_i^t R_j^t | i \neq j], \quad a = \sum_{k=1}^{\mu} (p_s(k))^2 \tag{7}$$

Symbol $p_s(k)$ stands for the selection probability of the k -th chromosome. Value of $E [(R^t)^2]$ can be computed by observing that each chromosome from R^t results from sampling with replacement from P^t . Hence the variance of points which are sampled from P^t for reproduction can be derived from the weighted variance formula:

$$E [(R^t)^2] = E \left[\sum_{k=1}^{\mu} (p_s(k))^2 (P_k^t)^2 \right] = aE[(P^t)^2] \tag{8}$$

Similar observations can be made for the covariance of points sampled from R^t :

$$E [R_i^t R_j^t | i \neq j] = E \left[2 \sum_{k=1}^{\mu-1} \sum_{l=k}^{\mu} (p_s(k)p_s(l)P_k^t P_l^t) \right] = bE [P_k^t P_l^t | k \neq l] \quad (9)$$

$$b = 2 \sum_{k=1}^{\mu-1} \sum_{l=k}^{\mu} p_s(k)p_s(l) \quad (10)$$

Values of $E[(P^t)^2]$ and $E[P_k^t P_l^t | k \neq l]$ are the weighted sample variance and covariance values with weights that are equal selection probabilities, hence

$$E[(P^t)^2] = (1 - a)v_P^t, \quad E[P_k^t P_l^t | k \neq l] = (1 - a)r_P^t \quad (11)$$

where v_P^t and r_P^t are the (theoretical) variance and covariance of a random variable which generates the population P^t . Then (7) can be rewritten as

$$r_P^{t+1} = a^2(1 - a)v_P^t + b(1 - a)^2 r_P^t \quad (12)$$

In the stable state it holds $r_P^{t+1} = r_P^t = r_P$ which allows to define r_P as

$$r_P = \frac{a^2(1 - a)}{1 - b(1 - a)^2} v_P \quad (13)$$

Equation (5) can be transformed to

$$V[\bar{P}^t] = \frac{1}{\mu'} v_P, \quad \mu' = \frac{1 - b(1 - a)^2}{1 - b(1 - a)^2 + a^2(1 - a)(\mu - 1)} \quad (14)$$

Value of μ' will be called the *effective population size* since it leads to formula for the variance of the population mean as if the chromosomes were independent. Then the empirical variance is chi-square distributed with the μ' degrees of freedom and its variance equals approximately

$$V[s^2(P^t)] \approx \frac{2(V[X])^2}{\mu' - 1} \quad (15)$$

Example. Above considerations are validated with the following experiment for the fitness function $q(x) = \exp(-x^2)$ and the EA with binary tournament selection, population size $\mu = 100$ and Gaussian mutation with variance $v_m = 1$. The EA was run 100 times, each run took $T = 10^4$ generations. Populations were initialized with standardized normal distribution. For each run, the series of states $u_t = [\bar{P}^t, s^2(P^t)]$, which describe dynamics of populations, was characterized with their mean and empirical variance:

$$\bar{u} = \left[\overline{\bar{P}^t}, \overline{s^2(P^t)} \right], \quad s^2(u) = [s^2(\bar{P}^t), s^2(s^2(P^t))] \quad (16)$$

For the binary tournament selection, values of a and b can be computed observing that $p_s(i) = (2\mu - 2i + 1)/(\mu^2)$ — see [1]:

$$a = \sum_{i=1}^{\mu} \left(\frac{2\mu - 2i + 1}{\mu^2} \right)^2 = \frac{4\mu^2 - 1}{3\mu^3}, \quad b = \frac{3\mu^4 - 8\mu^3 + 11\mu - 6}{3\mu^4}$$

which yields the approximate effective population size $\mu' = \frac{3}{4}\mu$.

Fig. 2 contains box-and-whisker plots of mean and variance of population empirical moments. Expected values of moments that have been predicted in previous section are given in corresponding plots. Mean relative error of prediction variance of moments equals 0.97% for $V[(\bar{P}^t)]$ and 5.58% for $V[s^2(P^t)]$. If independence of chromosomes in populations were assumed then the corresponding error values would equal 24% for $V[(\bar{P}^t)]$ and 21% for $V[s^2(P^t)]$.

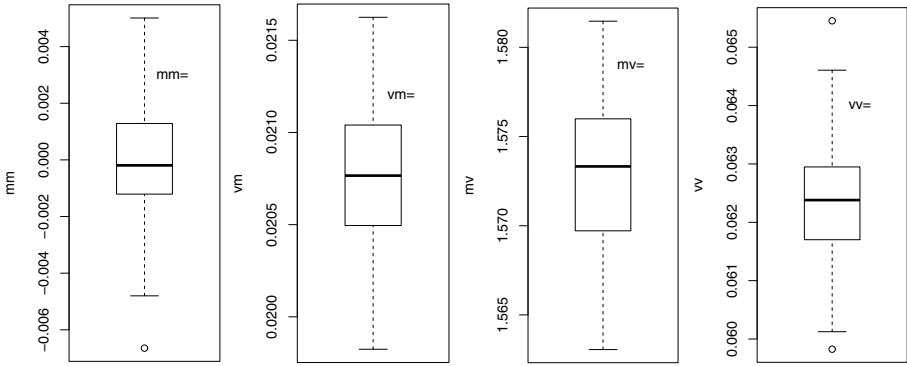


Fig. 2. Box and whisker plots of mean and variance values of population mean and variance

4 Quasi-Stability of Finite Populations

For a realistic EA it is impossible to expect its stability in a strict sense since each state is affected by mutation. For this reason it is impossible to find a combination of population mean and variance values which will not be changed in the next generation. The quasi-stability discussed here is defined as follows.

Consider a set of EA states $U(u^*, p)$, which is characterized by a state u^* , and a p-value p . The set $U(u^*, p)$ is quasi-stable when the condition is satisfied

$$\forall u_t \in U(u^*, p) \quad E[u_{t+1}] \in U(u^*, p) \tag{17}$$

where $E[u_{t+1}] = [E[\bar{P}^{t+1}], E[s^2(P^{t+1})]]$ is the vector of expected values of the next state. In other words, if the EA population is characterized by the state $u_t \in U(u^*, p)$ then the next state should not tend to leave the set $U(u^*, p)$.

Bounds on Quasi-Stable Population State. Equations (4), (14) and (15) can be used to define the set $U(u^*, p)$ by defining ranges of values of population mean and variance where their actual values can be found with certain probability. If the population distribution is normal then values of \bar{P}^t are normally distributed and $s^2(P^t)$ is chi-square distributed with $\mu' - 1$ degrees of freedom. For each population state $[\bar{P}^t, s^2(P^t)] \in U(u^*, p)$ it holds

$$m^* - \alpha(v^*, \mu', p) \leq \bar{P}^t \leq m^* + \alpha(v^*, \mu', p) \tag{18}$$

$$\beta(v^*, \mu', p) \leq s^2(P^t) \leq \gamma(v^*, \mu', p) \tag{19}$$

where p is the probability of observing values of population mean or variance outside the set $U(u^*, p)$. Values of α, β, γ are defined as upper and lower quantiles of normal and chi-square distributions for the probability $p/2$:

$$\alpha(v, \mu, p) = \sqrt{\frac{v}{\mu}} \cdot Q_n \left(1 - \frac{1}{2}p \right) \tag{20}$$

$$\beta(v, \mu, p) = \frac{v \cdot Q_c \left(\frac{1}{2}p, \mu - 1 \right)}{\mu - 1} \tag{21}$$

$$\gamma(v, \mu, p) = \frac{v \cdot Q_c \left(1 - \frac{1}{2}p, \mu - 1 \right)}{\mu - 1} \tag{22}$$

where $Q_n, Q_c(\cdot, k)$ represent the quantile generation functions for the normal and chi-square distribution with k degrees of freedom, respectively.

Testing Quasi-Stability for Finite Populations. For finite populations, if the population state $u_t = [\bar{P}^t, s^2(P^t)]$ is known then it is possible to predict expected values of the next population state, $E[u_{t+1}]$, without knowing the exact contents of P^t . The prediction is based on an assumption that P^{t+1} contains points which are generated randomly with expectation \bar{P}^t and variance $s^2(P^t)$:

$$E[\bar{P}^{t+1}] = \bar{P}^t + \sqrt{s^2(P^t)} \cdot \phi(u_t) \tag{23}$$

$$E[s^2(P^{t+1})] = s^2(P^t) \cdot \theta(u_t) + v_m \tag{24}$$

Symbol $\phi(u)$ is called the Response to Selection in Mean (RSM) and indicates the expected change of the population mean in effect of selection

$$\phi(u) = \frac{1}{\sqrt{v}} \int_{-\infty}^{\infty} (x - m)c(x)dx \tag{25}$$

where $c(x)$ stands for the p.d.f. of chromosomes reproduced from a population which is normally distributed with expectation m , variance v and density $g_{m,v}(x)$. Symbol $\theta(u)$ denotes the Response to Selection in Variance (RSV) which is the degree of the population variance reduction after selection

$$\theta(u) = \frac{1}{v} \int_{-\infty}^{\infty} (x - m)^2 c(x)dx \tag{26}$$

Note that RSM and RSV depend on selection since $c(x)$ depends on it.

Response to selection, which has been originally defined as the expected change in fitness of individuals before and after selection, is a concept which has been introduced to the evolutionary computation field by Muehlenbein [7] who adopted it from the breeding science in order to define the Breeder Genetic Algorithm. Here, instead of analyzing fitness of individuals, we concentrate on changes of mean and variance of chromosomes processed by the EA to analyze selection effects on the level of genotypes rather than fitness.

Equations (14), (18) and (19) allow for formulating the test of quasi-stability of a set of states. An area $U(u^*, p)$ is quasi-stable with mean m^* and variance v^* when for each population P characterized by the state $u = (\bar{P}, s^2(P)) \in U(u^*, p)$ it holds

$$-\alpha(v^*, \mu', p) \leq \bar{P} - m^* + \phi(u) \cdot \sqrt{v} \leq \alpha(v^*, \mu', p) \tag{27}$$

$$\beta(v^*, \mu', p) \leq \theta(u) \cdot s^2(P) + v_m \leq \gamma(v^*, \mu', p) \tag{28}$$

Example. Consider a fitness function which is defined as a composition of two Gaussian hills¹— see Fig. 3.

$$q(x) = 5 \exp(-x^2/2) + 4 \exp(-(x - 9)^2/8) \tag{29}$$

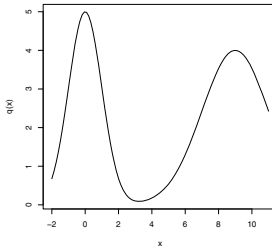


Fig. 3. Plot of the Galar function

Consider binary tournament selection. From [1] it follows that $c(x)$ is given by

$$c(x) = \int_{q(y) < q(x)} g_{m,v}(y) dy \cdot g_{m,v}(x) \tag{30}$$

which allows to define α, β and γ .

Typical dynamics of the population mean and variance of single run of an EA with $\mu = 100$ chromosomes which optimizes the Galar functions is illustrated in Fig. 4 for few characteristic values of the mutation variance v_m . In all cases the population was initialized with clones of the point 0 and the EA was run for 10^4 generations. In each plot a point represents state u_t of a population P^t and a rectangle is the set $U(u_P^*, p)$ which is a candidate for the quasi-stability area whose limits have been computed putting the stable variance prediction v_P

¹ This function was introduced by R.Galar and discussed e.g. in [8].

yielded by the infinite population model [1] into formulas (27), (28). Value m_P^* has been determined by solving the equation $\phi(u) = 0$ with respect to m , assuming that $v = v_P$.

When $v_m = 0.01$, states of populations stay, except in a few observations, in the quasi-stability rectangle that relates to the global maximum $m_P \approx 0$. For $v_m = 1.18$ the population state mean makes incidental “excursions” from this quasi-stability area towards states characterized with larger mean and variance, but then it returns.

When $v_m = 1.3$, the population state remains for a number of generations in the quasi-stability area around $m_P \approx 0$. Then the population state shifts towards the quasi-stability area corresponding to the second local maximum of the fitness at $m_P \approx 9$. There it remains stable for the rest of the simulation. Note that in this case, populations changed their position despite of the fact that the first quasi-stability area corresponds with the global maximum of the fitness function. This is an illustration of the “survival of the flattest” effect [9] which consists in preference to chromosomes whose fitness values are little sensitive to changes of their position in the chromosomes space.

For $v_m = 10$ a single quasi-stable area exists and then $m_P \approx 7$. Note that this value differs significantly from position of any local maximum. Note that although the assumption about unimodality of the fitness function is not satisfied, the population variance values, which have been predicted in [1], yield correct stability margins and properly explain quasi-stability of populations.

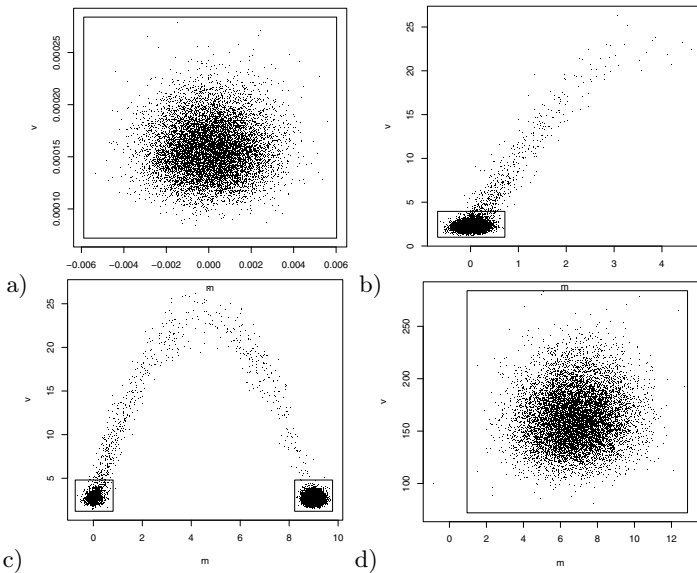


Fig. 4. Plots of the states of populations which were started at 0 and were run for 10^4 generations for the following mutation variance values: a) $v_m = 0.01$, b) $v_m = 1.18$, c) $v_m = 1.3$, d) $v_m = 10$; rectangles represent quasi-stability areas for p-value $p = 10^{-5}$

5 Closing Remarks

In many engineering applications stability of a system is usually a desired feature. For the EA, quasi-stability is a mixed blessing. On one hand, quasi-stability allows for better exploitation of area $U(u_P, p)$ which may contain some local optimum nearby its midpoint. On the other hand, if populations are quasi-stable for a very low p-value in an area that relates to an optimum of the fitness function, then it is hardly possible for the EA to switch to some other quasi-stability area that relates to another local optimum. Such quasi-stability is undesired since the resulting EA will be a poor global optimizer. This suggests that a method that tracks populations to detect quasi-stability and to break it may be considered as a yet another adaptation method.

It is interesting how the presented results will generalize in real space with many dimensions. Intuitively, response to selection functions are equivalents of derivatives and they may be generalized to a form of a “gradient” by computing the RSM and RSV values for each dimension separately. Then the quasi-stability margins would be defined for each dimension separately, provided that a proper infinite population model to predict the stable variance is developed. These directions of research define the scope of future work.

References

1. Arabas, J.: Approximating the genetic diversity of populations in the quasi-equilibrium state. *IEEE Transactions on Evolutionary Computation* 16(5), 632–644 (2012)
2. Beyer, H.G., Deb, K.: On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 5(3), 250–270 (2001)
3. Arnold, D.V., Beyer, H.G.: On the benefits of populations for noisy optimization. *Evolutionary Computation* 11(2), 111–127 (2003)
4. Arnold, D.V.: *Noisy Optimization with Evolution Strategies*. Kluwer Academic Publishers (2002)
5. Qi, X., Palmieri, F.: Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part I: Basic properties of selection and mutation. *IEEE Transactions on Neural Networks* 5(1), 102–119 (1994)
6. Karcz-Duleba, I.: Dynamics of infinite populations evolving in a landscape of uni and bimodal fitness functions. *IEEE Transactions on Evolutionary Computation* 5(4), 398–409 (2001)
7. Muehlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the Breeder Genetic Algorithm – I. continuous parameter optimization. *Evolutionary Computation* 1, 25–49 (1993)
8. Chorazyczewski, A., Galar, R.: Visualization of evolutionary adaptation in R^n . In: Porto, V.W., Waagen, D. (eds.) *EP 1998*. LNCS, vol. 1447, pp. 659–668. Springer, Heidelberg (1998)
9. Wilke, C., et al.: Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature* 412(6844), 331–333 (2001)

On the Use of Evolution Strategies for Optimization on Spherical Manifolds

Dirk V. Arnold

Faculty of Computer Science, Dalhousie University
Halifax, Nova Scotia, B3H 4R2, Canada
`dirk@cs.dal.ca`

Abstract. We study the behaviour of evolution strategies applied to a simple class of unimodal optimization problems on spherical manifolds. The techniques used are the same as those commonly employed for the analysis of the behaviour of evolution strategies in Euclidean search spaces. However, we find that there are significant differences in strategy behaviour unless the vicinity of an optimal solution has been reached. Experiments with cumulative step size adaptation reveal the existence of metastable states associated with large step sizes, which can preclude reaching optimal solutions.

1 Introduction

The vast majority of work on real-valued evolutionary optimization is concerned with Euclidean search spaces. However, there are important applications where the search domain is not Euclidean but a more general Riemannian manifold instead. See [6] for an introduction to Riemannian geometry. Qi et al. [10] give two broad classes of applications for optimization on manifolds: “equality-constrained optimization problems where the constraints specify a submanifold of \mathbb{R}^N ; and problems where the objective function has continuous invariance properties that we want to eliminate for various reasons”. Our own interest in optimization on manifolds is motivated by the need to optimize quaternion variables, which commonly arises in 3D registration tasks where quaternion variables are used to encode orientation.

Several applications of evolutionary algorithms to optimization on Riemannian manifolds other than Euclidean spaces can be found in the literature. An early instance is work by Kissinger et al. [9], who propose a variant of evolutionary programming for optimization involving quaternion variables. Arguably the most sophisticated evolutionary approach to optimization on general Riemannian manifolds is that by Colutto et al. [5], who propose a variant of covariance matrix adaptation evolution strategies (CMA-ES) [7] for optimization on Riemannian manifolds. Their algorithm uses parallel transport, a tool for transporting geometrical data along smooth curves, to transform search paths and covariance matrices between iterations. They find that their approach more effectively solves a two-dimensional multimodal optimization problem than restart

variants of gradient based and Newton-Armijo methods proposed by Yang [12] for optimization on manifolds. However, they also observe that their strategy can (infrequently) be observed to fail to locate the optimal solution to a simple unimodal optimization problem on spherical manifolds.

The aim of this paper is to develop an analytically based understanding of the behaviour of a simplified variant of the algorithm of Colutto et al. [5] that adapts its global step size using cumulative step size adaptation, but does not adapt the full covariance matrix. We derive results characterizing the behaviour of the algorithm when applied to a class of unimodal optimization problems on spherical manifolds, both providing an explanation for the observed inability to converge to the optimal solution in some instances and suggesting how to avoid this situation.

2 Problem and Algorithm

Let M be a Riemannian manifold with tangent space $T_p M$ at point $p \in M$. We consider the case that $M = S^{N-1}$ (i.e., the unit $(N-1)$ -sphere defined by $S^{N-1} = \{\mathbf{x} \in \mathbb{R}^N \mid \|\mathbf{x}\| = 1\}$). Note that the universal cover of the 3D rotation group $SO(3)$ is diffeomorphic to S^3 , and that problems on spherical manifolds thus naturally arise in combination with the optimization of orientations in 3D. We consider the class of optimization problems $f : S^{N-1} \rightarrow \mathbb{R}$ that possess a unique optimal solution and where objective function values depend only on the distance from that solution and vary strictly monotonically with it. By choosing a coordinate system such that the optimal solution is located at $\mathbf{x} = (1, 0, \dots, 0)^T$ and taking into account that evolution strategies perform selection based only on comparisons of objective function values, we can without loss of generality consider objective function

$$f(\mathbf{x}) = x_1 \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \dots, x_N)^T \in S^{N-1}$ and the task is maximization. Effectively the same problem has been used by Colutto et al. [5] in the experimental evaluation of their algorithm. Arguably, Eq. (1) constitutes the analogue of the “sphere model” introduced by Rechenberg [11] for the study of the behaviour of evolution strategies in Euclidean spaces.

The CMA-ES for optimization on Riemannian manifolds proposed by Colutto et al. [5] generates mutation vectors in the tangent space at the current population centroid, and it uses the Riemannian exponential map to map them onto the manifold. Parallel transport is used as a means for mapping search paths and the covariance matrix of the mutation distribution from the tangent space at the current population centroid to that at the next. The algorithm considered here is in essence the same, with the single major difference that in order to admit an analytically based investigation, rather than adapting the entire covariance matrix, only the global step size is adapted. A single iteration of the algorithm is given in Fig. 1. The population size parameters μ and λ are positive integers with $\mu < \lambda$. Cumulation parameter $c \in (0, 1]$ and damping parameter $D \in \mathbb{R}^+$

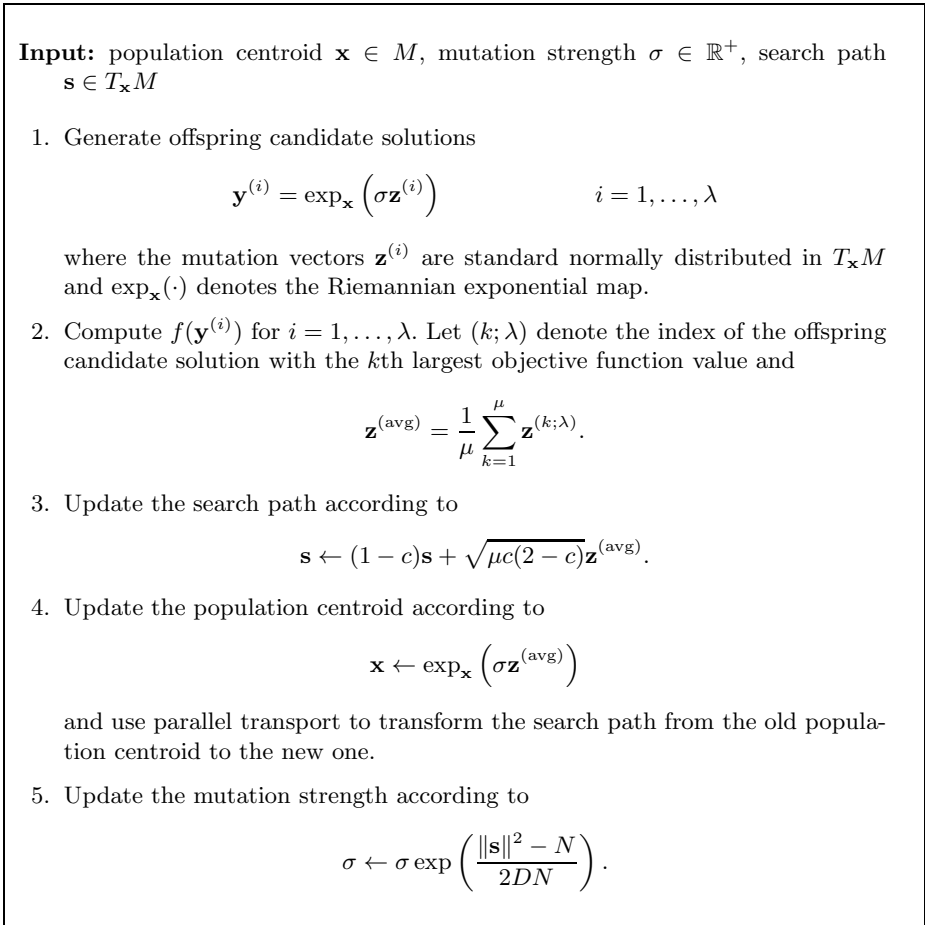


Fig. 1. Single iteration of the strategy for optimization on Riemannian manifolds

are constants. After initialization (to be discussed below), the algorithm in Fig. 1 is iterated until a stopping condition is met.

Regarding the implementation of the algorithm for the case that $M = S^{N-1}$, sampling standard normally distributed mutation vectors in $T_{\mathbf{x}}S^{N-1}$ can be accomplished by sampling standard normally distributed mutation vectors \mathbf{w} in \mathbb{R}^N and projecting them onto the tangent space $T_{\mathbf{x}}S^{N-1}$ according to

$$\mathbf{z} = \mathbf{w} - \langle \mathbf{x}, \mathbf{w} \rangle \mathbf{x} \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. The Riemannian exponential map $\exp_{\mathbf{x}}(\cdot) : T_{\mathbf{x}}M \rightarrow M$ for the case that $M = S^{N-1}$ is described by

$$\exp_{\mathbf{x}}(\sigma \mathbf{z}) = \mathbf{x} \cos(\sigma \|\mathbf{z}\|) + \mathbf{z} \frac{\sin(\sigma \|\mathbf{z}\|)}{\|\mathbf{z}\|}. \tag{3}$$

Finally, Huckemann et al. [8] show that parallel transport on spherical manifolds is accomplished for $\mathbf{x} \neq \pm \mathbf{x}'$ by

$$\mathbf{w}' = \mathbf{w} - \langle \mathbf{w}, \mathbf{v} \rangle [(1 - \langle \mathbf{x}, \mathbf{x}' \rangle) \mathbf{v} + \langle \mathbf{v}, \mathbf{x}' \rangle \mathbf{x}] \tag{4}$$

where

$$\mathbf{v} = \frac{\mathbf{x}' - \langle \mathbf{x}, \mathbf{x}' \rangle \mathbf{x}}{\|\mathbf{x}' - \langle \mathbf{x}, \mathbf{x}' \rangle \mathbf{x}\|} \tag{5}$$

and \mathbf{w}' is the parallel transplant of $\mathbf{w} \in T_{\mathbf{x}}S^{N-1}$ to $T_{\mathbf{x}'}S^{N-1}$.

3 Analysis

In order to analyze the behaviour of the algorithm thus described when applied to the class of problems defined by Eq. (1) we first consider single iterations and determine the expected step. In analogy to related work in Euclidean spaces [11, 2] we then obtain simpler expressions by making the assumption that the steps that the strategy takes are small. It will be seen that that assumption is a valid one to make only if the strategy has reached the vicinity of the optimal solution. Finally, we consider the multi-iteration behaviour of the algorithm.

3.1 Large-Step Behaviour

Considering a single iteration of the algorithm described in Fig. 1, by choosing the coordinate system appropriately we can without loss of generality assume that $\mathbf{x} = (x_1, \sqrt{1 - x_1^2}, 0, \dots, 0)^T$. From Eq. (1) with Eq. (3), the objective function value of offspring candidate solution $\mathbf{y} = \exp_{\mathbf{x}}(\sigma \mathbf{z})$ is

$$f(\mathbf{y}) = x_1 \cos(\sigma \|\mathbf{z}\|) + z_1 \frac{\sin(\sigma \|\mathbf{z}\|)}{\|\mathbf{z}\|}. \tag{6}$$

The lengths $\|\mathbf{z}\|$ of mutation vectors are χ_{N-1} -distributed with mean $l_{\mathbf{z}} = \sqrt{2}\Gamma(N/2)/\Gamma((N-1)/2)$, which for large N is well approximated by \sqrt{N} , and a coefficient of variation that goes to zero as N increases. The impact of variations in $\|\mathbf{z}\|$ on offspring objective function values thus decreases with increasing dimension N . The ordering of the offspring by objective function values in Step 2 of the algorithm in Fig. 1 will thus increasingly be an ordering by values of z_1 . From Eq. (2), $z_1 = (1 - x_1^2)w_1 - \sqrt{1 - x_1^2}x_1w_2$, where w_1 and w_2 are independently standard normally distributed. Mutation vector component z_1 is thus normally distributed with mean zero and variance $1 - x_1^2$. The selected z_1 -components are the μ largest order statistics of a sample of normally distributed random variates, and their expected average according to [2] equals

$$\mathbb{E} \left[z_1^{(\text{avg})} \right] = \text{sgn}(\sin(\sigma \|\mathbf{z}\|)) \sqrt{1 - x_1^2} c_{\mu/\mu, \lambda} \tag{7}$$

where $c_{\mu/\mu, \lambda}$ denotes the progress coefficient. Defining the progress rate

$$\varphi = \mathbb{E} \left[f \left(\exp_{\mathbf{x}} \left(\sigma \mathbf{z}^{(\text{avg})} \right) \right) \right] - f(\mathbf{x}) \tag{8}$$

as the expected improvement in objective function value in a single iteration, it follows from Eq. (7) that for large N

$$\varphi = \text{sgn}(\sin(\sigma\|\mathbf{z}\|)) \frac{\sin(\sigma\|\mathbf{z}^{(\text{avg})}\|)}{\|\mathbf{z}^{(\text{avg})}\|} \sqrt{1 - x_1^2} c_{\mu/\mu,\lambda} - \left[1 - \cos(\sigma\|\mathbf{z}^{(\text{avg})}\|)\right] x_1 \quad (9)$$

approximately holds. As $\mathbf{z}^{(\text{avg})}$ is the average of μ vectors $N - 2$ of the $N - 1$ components of which are random and uncorrelated, for large N the expected length of $\mathbf{z}^{(\text{avg})}$ can be approximated by $l_{\mathbf{z}}/\sqrt{\mu}$ [2].

The left hand side of Fig. 2 compares predictions from Eq. (9) with $\|\mathbf{z}\|$ and $\|\mathbf{z}^{(\text{avg})}\|$ replaced with their expected values with measurements made in one-iteration experiments of the algorithm for several values of N . Values of $x_1 \in \{-0.5, 0.2, 0.9\}$ have been chosen as representatives of situations where the search is at a great distance, an intermediate distance, and in relative proximity to the optimal solution and thus of different stages in the optimization process. Clearly, the accuracy of the predictions improves with increasing N and decreases with increasing σ . The primary reason for the inaccuracies are variations in the length of mutation vectors. In low dimensions and for large mutation strengths, selection is increasingly on the basis of the length of the mutation vectors. Nonetheless, it is clear from the figure that the dependence of the progress rate on the mutation strength differs markedly from that for the Euclidean sphere model, where the progress rate after an initial increase monotonically decreases to negative infinity [11, 2]. On the spherical manifold, multiple modes can be observed that result from the Riemannian exponential map tracing out geodesic paths. Additionally, the curves exhibit discontinuities that stem from the lengths of the average of the selected mutation vectors being reduced compared to those of the mutation vectors themselves. Mutation vectors of a length resulting in the sine function in Eq. (6) being positive may result in negative values of the sine function in the numerator in Eq. (9) and vice versa.

3.2 Small-Step Behaviour

All of the curves in the graphs on the left hand side of Fig. 1 have in common that for small mutation strengths they initially increase. More often than never, the first mode encountered yields optimal or near optimal performance. To investigate the behaviour of the strategy for small mutation strengths, we replace $\|\mathbf{z}\|$ and $\|\mathbf{z}^{(\text{avg})}\|$ in Eq. (9) with \sqrt{N} and $\sqrt{N/\mu}$, respectively, expand the sine and cosine functions into Taylor series at zero and abort after the linear and quadratic terms, respectively, and for $x_1 \neq \pm 1$ introduce normalized mutation strength $\sigma^* = N|x_1|\sigma/\sqrt{1 - x_1^2}$ and normalized progress rate $\varphi^* = N|x_1|\varphi/(1 - x_1^2)$, resulting in

$$\varphi^* = \sigma^* c_{\mu/\mu,\lambda} - \text{sgn}(x_1) \frac{\sigma^{*2}}{2\mu} . \quad (10)$$

Notice that for $x_1 > 0$ Eq. (10) has the same form as the progress rate law for the Euclidean sphere model [2].

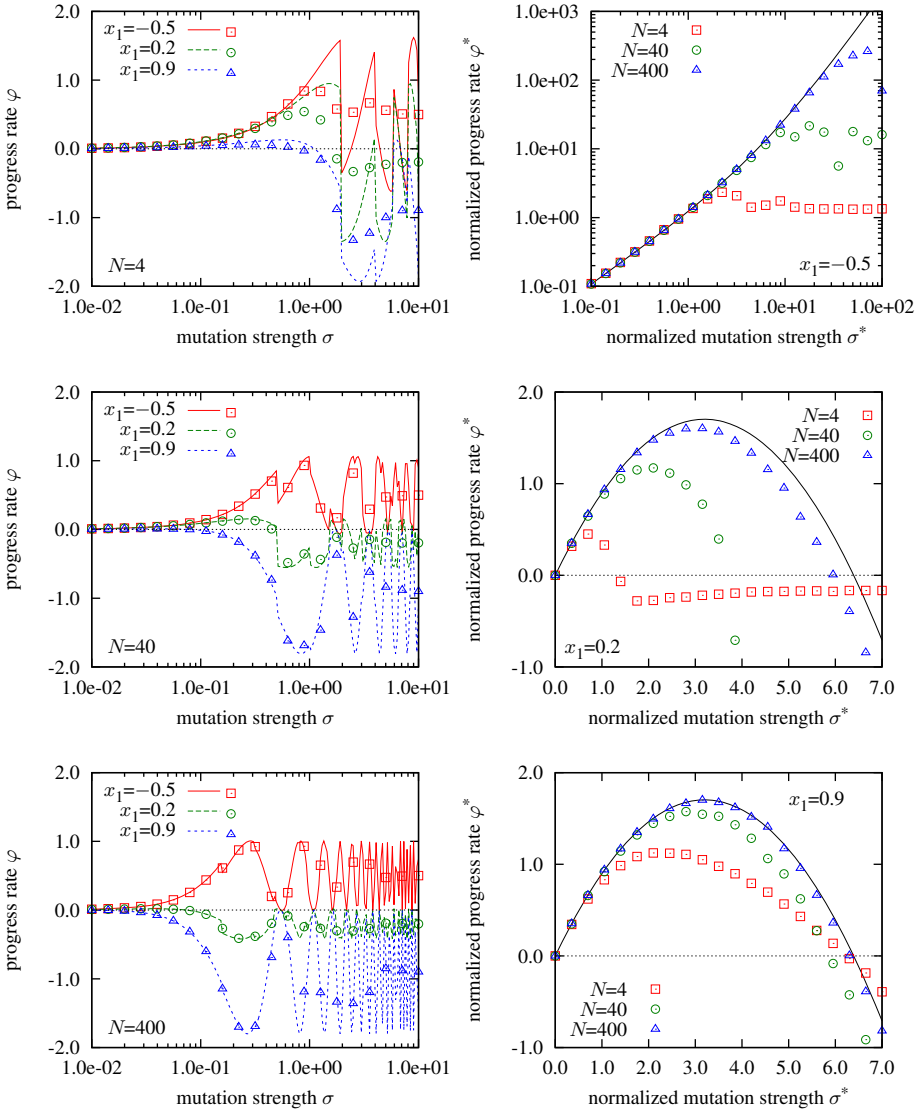


Fig. 2. Left: Progress rate φ plotted against mutation strength σ for $\mu = 3$ and $\lambda = 10$, parent locations $x_1 \in \{-0.5, 0.2, 0.9\}$, and, from top to bottom, search space dimensions $N = 4, 40$, and 400 . Right: Normalized progress rate φ^* plotted against normalized mutation strength σ^* for search space dimensions $N \in \{4, 40, 400\}$ and, from top to bottom, parent locations $x_1 = -0.5, 0.2$, and 0.9 . The lines represent predictions from Eqs. (9) and (10), respectively. The points mark values obtained by averaging over 20,000 one-iteration experiments for each data point shown.

The right hand side of Fig. 2 compares predictions from Eq. (10) with measurements made in one-iteration experiments of the algorithm. For $x_1 = -0.5$ Eq. (10) predicts that the normalized progress rate increases indefinitely with increasing normalized mutation strength. This is of course impossible for finite N , and the solid curve fails to correctly predict the experimental data if the mutation strength is too large for the truncated Taylor series to well represent the trigonometric functions. However, as increasing σ results in tracing out a geodesic path, the hemisphere with $x_1 > 0$ can always be reached in a single step. For $x_1 = 0.2$ the accuracy of the predictions increases significantly with increasing N . The truncated Taylor series become poor approximations to the trigonometric functions for $\sigma\|\mathbf{z}\| \approx \pi/2$ and thus for $\sigma^* \approx \pi|x_1|\sqrt{N}/(2\sqrt{1-x_1^2})$, and the maxima of the experimental data for $N = 4$ and $N = 40$ quite closely correspond to those values. Finally, for $x_1 = 0.9$ the qualitative behaviour of the algorithm is captured quite well for N as small as 4. If the mutation strength of the strategy is controlled properly, values of x_1 in excess of 0.9 can be reached in a relatively small number of iterations, and much of the computational effort will be incurred where the predictions from Eq. (10) are quite accurate.

3.3 Step Size Adaptation

To analyze the performance of cumulative step size adaptation on spherical manifolds, we employ the same approach as in Euclidean spaces [1]. The state of the strategy is determined by the population centroid \mathbf{x} , the mutation strength σ , and the search path \mathbf{s} . The parallel transport in Step 4 of the algorithm Fig. 1 uses vector \mathbf{v} , which can be computed from Eq. (5) as $\mathbf{z}^{(\text{avg})}/\|\mathbf{z}^{(\text{avg})}\|$. Using primes to indicate values of a quantity after an iteration of the algorithm, it follows that $\langle \mathbf{x}, \mathbf{x}' \rangle = \cos(\sigma\|\mathbf{z}^{(\text{avg})}\|)$ and $\langle \mathbf{v}, \mathbf{x}' \rangle = \sin(\sigma\|\mathbf{z}^{(\text{avg})}\|)$. Omitting terms that disappear in the limit $N \rightarrow \infty$ and using the small-step approximation from Sect. 3.2, the update of the search path in Steps 3 and 4 of the algorithm in Fig. 1 is thus described by

$$\mathbf{s}' = (1 - c)\mathbf{s} + \sqrt{\mu c(2 - c)} \left[\mathbf{z}^{(\text{avg})} \cos(\sigma\|\mathbf{z}^{(\text{avg})}\|) - \mathbf{x}\|\mathbf{z}^{(\text{avg})}\| \sin(\sigma\|\mathbf{z}^{(\text{avg})}\|) \right] \tag{11}$$

where it is assumed that $c = 1/\sqrt{N}$ and $D = 1/c$.¹

Due to the symmetry inherent in the problem at hand, the location of the population centroid is adequately described by x_1 , and the search path is characterized by its components s_1 and $s_\odot = \sum_{i=2}^N x_i s_i / \sqrt{\sum_{i=2}^N x_i^2}$, along with its squared length $\|\mathbf{s}\|^2$. Iterating the algorithm in Fig. 1 generates a Markov process in a five-dimensional state space with variables x_1 , σ , s_1 , s_\odot , and $\|\mathbf{s}\|^2$. We compute an approximation to the average values characterizing the search path

¹ Detailed calculations cannot be reproduced here due to space limitations, but can be found at <http://www.cs.dal.ca/~dirk/PPSN2014addendum.pdf> instead.

by requiring that an iteration of the algorithm results in no change in expectation. That is, we require that $E[s'_1] = s_1$, $E[s'_\odot] = s_\odot$, and $E[\|s'\|^2] = \|s\|^2$. Dropping terms that disappear for large N and solving for $\|s\|^2$ yields

$$\|s\|^2 = N + 2 \frac{\mu c_{\mu/\mu,\lambda}}{c} \left[c_{\mu/\mu,\lambda} - \text{sgn}(x_1) \frac{\sigma^*}{\mu} \right] \tag{12}$$

for the squared length of the search path.

As in [1] we refer to the mutation strength for which no change in step size is expected as the target mutation strength of the strategy. For $x_1 > 0$, from the update of the mutation strength in Step 5 of the algorithm in Fig. 1 with Eq. (12), the target mutation strength is $\sigma^*_{\text{target}} = \mu c_{\mu/\mu,\lambda}$, which is optimal according to Eq. (10). However, the normalized mutation strength actually attained by the strategy differs from the target mutation strength as the distance from the optimal solution decreases simultaneously with the step size and adaptation is not instantaneous. Calculations equivalent to those in [1] yield

$$\sigma^* = \mu c_{\mu/\mu,\lambda} \left[(1 - x_1^2) \text{sgn}(x_1) + \sqrt{1 + x_1^4} \right] \tag{13}$$

for the mutation strength attained by the strategy. That is, normalized mutation strengths generated by cumulative step size adaptation for $x_1 \lesssim 1$ exceed optimal ones by a factor of $\sqrt{2}$, resulting in a 17% loss of performance (compare [3]). For $x_1 \gtrsim 0$ Eq. (13) suggests that mutation strengths generated by cumulative step size adaptation are nearly twice as large as optimal, resulting in near zero progress and thus stagnation of the strategy. However, as seen above, the small-step predictions are highly inaccurate for $x_1 \approx 0$ and the validity of the findings needs to be confirmed experimentally.

Figure 3 shows partial traces from typical runs of the evolution strategy for different search space dimensions. For each combination of parameter settings, 99 runs were conducted until either a solution with an objective function value within 10^{-6} of optimal was generated or 20,000 iterations were reached. The runs shown in the figure are those with the median number of iterations until termination, where ties were broken arbitrarily. Each run was initialized with $x_1 = 0$ and an initial mutation strength $\sigma_0 \in \{0.1, 1.0, 10.0\}$. It can be seen that the behaviour of the algorithm depends qualitatively on the initial mutation strength. Too large a value of σ_0 (where what is “too large” depends on N) results in the strategy operating past the first mode observed in Fig. 2. Cumulative step size adaptation in that situation either does not decrease the step size or decreases it only very slowly. The graphs on the right hand side of Fig. 3 show that the strategy in this situation jumps about apparently symmetrically about $x_1 = 0$, without reaching a point in the vicinity of the optimal solution. It can also be seen that once a point in the vicinity of the optimal solution is reached, cumulative step size adaptation controls the mutation strength as expected, and Eq. (13) quite closely predicts the further behaviour of the strategy. The observed metastable states characterized by relatively large mutation strengths and expected x_1 values of zero are more easily broken out of for small values of N , where the variance of the observed x_1 values is larger.

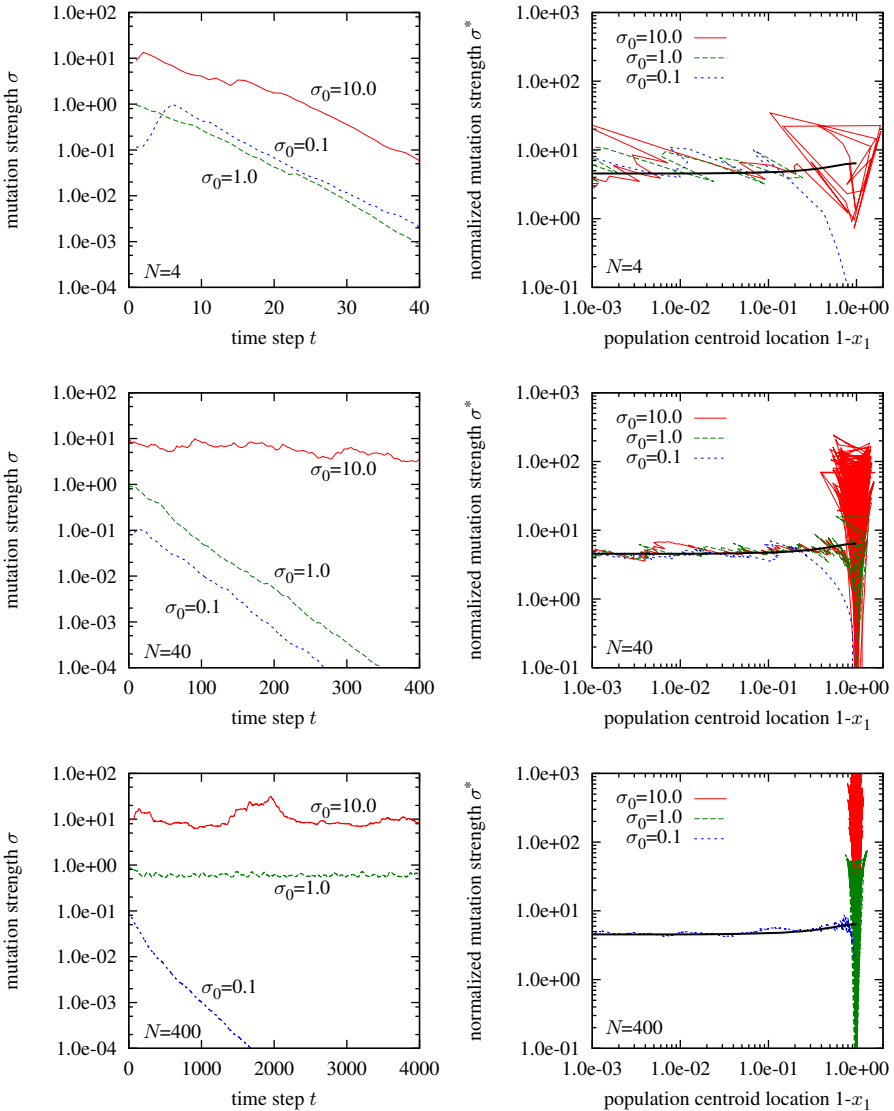


Fig. 3. Left: Mutation strength σ generated using cumulative step-size adaptation plotted against iteration number t for typical runs with $\mu = 3$ and $\lambda = 10$, initial step sizes in $\{0.1, 1.0, 10.0\}$, and, from top to bottom, search space dimensions $N = 4, 40,$ and 400 . Right: Normalized mutation strength σ^* from the same runs as shown on the left plotted against the transformed location $1 - x_1$ of the population centroid. The bold solid lines on the right hand side represent predictions from Eq. (13).

4 Discussion

We have presented a small-step approximation to the behaviour of a simplified variant of the algorithm of Colutto et al. [5] applied to a class of unimodal optimization problems on spherical manifolds. The approximation quite accurately describes the behaviour of the strategy in the vicinity of the optimal solution, but it is insufficient as a model in greater distance from that solution. In the latter case, large mutation strengths can result in the strategy operating in a metastable state rather than converging to the optimal solution. The analysis also suggests an approach for avoiding such metastable states: limiting the step size to at most $\sigma \approx \pi/(2\sqrt{N})$ ensures that the strategy does not operate significantly past the first mode in Fig. 2 and effectively prevents the long periods of stagnation observed in Fig. 3. In future work, we will attempt to derive an equivalent cap in the case of general Riemannian manifolds and compare the performance of the resulting algorithm with that of the approaches in *Manopt* [4].

References

- [1] Arnold, D.V., Beyer, H.-G.: Performance analysis of evolutionary optimization with cumulative step length adaptation. *IEEE Transactions on Automatic Control* 49(4), 617–622 (2004)
- [2] Beyer, H.-G.: *The Theory of Evolution Strategies*. Springer (2001)
- [3] Beyer, H.-G., Arnold, D.V.: Qualms regarding the optimality of cumulative path length control in CSA/CMA-evolution strategies. *Evolutionary Computation* 11(1), 19–28 (2003)
- [4] Boumal, N., Mishra, B., Absil, P.-A., Sepulchre, R.: *Manopt*, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research* 15, 1455–1459 (2014)
- [5] Colutto, S., Frühauf, F., Fuchs, M., Scherzer, O.: The CMA-ES on Riemannian manifolds to reconstruct shapes in 3-D voxel images. *IEEE Transactions on Evolutionary Computation* 14(2), 227–245 (2010)
- [6] do Carmo, M.P.: *Riemannian Geometry*. Birkhäuser (1992)
- [7] Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
- [8] Huckemann, S., Hotz, T., Munk, A.: Intrinsic MANOVA for Riemannian manifolds with an application to Kendall’s space of planar shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(4), 593–603 (2010)
- [9] Kissinger, C.R., Gehlhaar, D.K., Fogel, D.B.: Rapid automated molecular replacement by evolutionary search. *Acta Crystallographica D* 55, 484–491 (1999)
- [10] Qi, C., Gallivan, K.A., Absil, P.-A.: Riemannian BFGS algorithm with applications. In: Diehl, M., et al. (eds.) *Recent Advances in Optimization and its Applications in Engineering*, pp. 183–192. Springer (2010)
- [11] Rechenberg, I.: *Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Friedrich Frommann Verlag (1973)
- [12] Yang, Y.: Globally convergent optimization algorithms on Riemannian manifolds: Uniform framework for unconstrained and constrained optimization. *Journal of Optimization Theory and Applications* 132(2), 245–265 (2007)

Unbiased Black-Box Complexity of Parallel Search

Golnaz Badkobeh¹, Per Kristian Lehre², and Dirk Sudholt¹

¹ University of Sheffield, UK

² University of Nottingham, UK

Abstract. We propose a new black-box complexity model for search algorithms evaluating λ search points in parallel. The parallel unbiased black-box complexity gives lower bounds on the number of function evaluations *every* parallel unbiased black-box algorithm needs to optimise a given problem. It captures the inertia caused by offspring populations in evolutionary algorithms and the total computational effort in parallel metaheuristics. Our model applies to all unary variation operators such as mutation or local search. We present lower bounds for the Leading-Ones function and general lower bound for all functions with a unique optimum that depend on the problem size and the degree of parallelism, λ . The latter is tight for OneMax; we prove that a $(1+\lambda)$ EA with adaptive mutation rates is an optimal parallel unbiased black-box algorithm.

1 Introduction

Black-box optimisation describes a challenging realm of problems where no algebraic model or gradient information is available. The problem is regarded a black box, and knowledge about the problem in hand can only be obtained by evaluating candidate solutions. General-purpose metaheuristics like evolutionary algorithms, simulated annealing, ant colony optimisers, tabu search, and particle swarm optimisers are well suited for black-box optimisation as they generally work well without any problem-dependent knowledge.

A lot of research has focussed on designing powerful metaheuristics, yet it is often unclear which search paradigm works best for a particular problem class, and whether and how better performance can be obtained by tailoring a search paradigm to the problem class in hand.

The black-box complexity of search algorithms captures the difficulty of problem classes in black-box optimisation. It describes the minimum number of function evaluations that *every* black-box algorithm needs to make to optimise a problem from a given class. It provides a rigorous theoretical foundation through capturing limits to the efficiency of all black-box search algorithms, providing a baseline for performance comparisons across all known and future metaheuristics as well as tailored black-box algorithms. Also it prevents algorithm designers from wasting effort on trying to achieve impossible performance.

The first black-box complexity model by Droste et al. [6] makes no restriction on the black-box algorithm. This leads to some unrealistic results, such

as polynomial black-box complexities of NP-hard problems [6]. Subsequent research introduced refined models that restrict the power of black-box algorithms, leading to more realistic results [4–6, 18]. Lehre and Witt introduced the unbiased black-box model [13] where black-box algorithms may only use operators without a search bias (see Section 2). This model initially considered unary operators (such as mutation) and was later extended to higher arity operators (such as crossover) [3] and more general search spaces [17]. It also led to the discovery of more efficient EA variants [2].

A shortcoming of the above models is that they do not capture the implicit or explicit parallelism at the heart of many common search algorithms. Evolutionary algorithms (EAs) such as $(\mu+\lambda)$ EAs or (μ,λ) EAs generate λ offspring in parallel. Using a large offspring population in many cases can decrease the number of generations needed to find an optimal solution¹. However, the number of function evaluations may increase as evolution can only act on information from the previous generation. A large offspring population can lead to inertia that slows down the optimisation process. Existing black-box models are unable to capture this inertia as they assume all search points being created in sequence.

The same goes for parallel metaheuristics such as island models evolving multiple populations in parallel (see, e. g. [14]). Parallelisation can decrease the number of generations, or parallel time. But the overall computational effort, the number of function evaluations across all islands, may increase. Lässig and Sudholt [11] used the following notion. Let T_λ be the random number of generations an island model with λ islands (each creating one offspring) needed to find a global optimum for a given problem. If using λ islands can decrease the parallel time by a factor of order λ , compared to just one island, $\lambda \cdot E(T_\lambda) = O(E(T_1))$, this is called a *linear speedup* (with regards to the parallel time, the number of generations). A linear speedups means that the total number of function evaluations, $\lambda \cdot E(T_\lambda)$, does not increase beyond a constant factor.

Recent work [11, 12, 15] considered illustrative problems from pseudo-Boolean optimisation and combinatorial optimisation, showing sufficient conditions for linear speedups. However, the absence of matching lower bounds makes it impossible to determine exactly for which parameters λ linear speedups are achieved.

We provide a parallel black-box model that captures and quantifies the inertia caused by offspring populations of size λ and parallel EAs evaluating λ search points in parallel. We present lower bounds on the black-box complexity for the well known LO problem and for the general class of functions with a unique optimum, revealing how the number of function evaluations increases with the problem size n and the degree of parallelism, λ . The results complement existing upper bounds [11], allowing us to characterise the realm of linear speedups, where parallelisation is effective.

Our lower bound for functions with a unique optimum is asymptotically tight: we show that for the ONEMAX problem, a $(1+\lambda)$ EA with an adaptive mutation rate is an optimal parallel unbiased black-box algorithm. Adaptive mutation

¹ This does not hold for all problems; De Jong, Jansen, and Wegener constructed problems where offspring populations drastically increase the number of generations [9].

rates decrease the expected running time by a factor of $\ln \ln \lambda$, compared to the $(1+\lambda)$ EA with the standard mutation rate $1/n$ (see He, Chen, and Yao [7]).

2 A Parallel Black-Box Model

Following Lehre and Witt [13], we only use unary unbiased variation operators, i. e., operators creating a new search point out of one search point. This includes local search, mutation in evolutionary algorithms, but it does not include recombination. Unbiasedness means that there is no bias towards particular regions of the search space; in brief, for $\{0, 1\}^n$, unbiased operators must treat all bit values 0, 1 and all bit positions $1, \dots, n$ symmetrically (see [13, 17] for details). This is the case for many common operators such as standard bit mutation.

Unbiased black-box algorithms query new search points based on the past history of function values, using unbiased variation operators. We define a *λ -parallel unbiased black-box algorithm* in the same way, with the restriction that in each round λ queries are made in parallel (see Algorithm 1). These λ queries only have access to the history of evaluations from previous rounds; they cannot access information from queries made in the same round. We refer to these λ search points as *offspring* to indicate search points created in the same round.

Algorithm 1. λ -parallel unbiased black-box algorithm

1. Let $t := 0$. Choose $x^1(0), \dots, x^\lambda(0)$ uniformly at random, compute $f(x^1(0)), \dots, f(x^\lambda(0))$, and let $I(0) := \{f(x^1(0)), \dots, f(x^\lambda(0))\}$.
 2. **repeat**
 3. **for** $1 \leq i \leq \lambda$ **do**
 4. Choose an index $0 \leq j \leq t$ according to $I(t)$.
 5. Choose an unbiased variation operator $p_v(\cdot \mid x(j))$ according to $I(t)$.
 6. Generate $x^i(t+1)$ according to p_v .
 7. **end for**
 8. **for** $1 \leq i \leq \lambda$ **do**
 9. Compute $f(x^i(t))$ and let $I(t) := I(t) \cup \{f(x^i(t))\}$.
 10. **end for**
 11. Let $t := t + 1$.
 12. **until** termination condition met
-

This black-box model includes offspring populations in evolutionary algorithms, for example $(\mu+\lambda)$ EAs or (μ, λ) EAs (modulo minor differences in the initialisation). It can further model parallel evolutionary algorithms such as cellular EAs with λ cells, or island models with λ islands, each of which generates one offspring in each generation.

The *unbiased black-box complexity* (*uBBC*) of a function class \mathcal{F} is the minimum worst-case runtime among all unbiased black-box algorithms [13] (equivalent to Algorithm 1 with $\lambda = 1$). The *unbiased λ -parallel black-box complexity* (*λ -upBBC*) of a function class \mathcal{F} is defined as the minimum worst-case number

of function evaluations among all unbiased λ -parallel algorithms satisfying the framework of Algorithm 1.

With increasing λ access to previous queries becomes more and more restricted. It is therefore not surprising that the black-box complexity is non-decreasing with growing λ . For every family of function classes \mathcal{F}_n and all $\lambda \in \mathbb{N}$,

$$\text{uBBC}(\mathcal{F}_n) = 1\text{-upBBC}(\mathcal{F}_n) \leq 2\text{-upBBC}(\mathcal{F}_n) \leq 3\text{-upBBC}(\mathcal{F}_n) \dots \quad (1)$$

$$\text{and } \text{uBBC}(\mathcal{F}_n) \leq \lambda\text{-upBBC}(\mathcal{F}_n) \leq \lambda \cdot \text{uBBC}(\mathcal{F}_n) \quad (2)$$

as any unbiased algorithm can be simulated by a λ -parallel unbiased black-box algorithm using one query in each round. Due to (1), there is a *cut-off point*

$$\lambda^* := \sup\{\lambda \mid \exists c > 0, n_0 \forall n \geq n_0 : \lambda\text{-upBBC}(\mathcal{F}_n) \leq c \cdot \text{uBBC}(\mathcal{F}_n)\}$$

such that c is a constant and for all $\lambda \leq \lambda^*$ the λ -parallel unbiased black-box complexity of \mathcal{F} is asymptotically equal to the regular unbiased black-box complexity. In this realm, parallelisation is most effective as the number of function evaluations does not increase (beyond constant factors). The number of rounds for an optimal black-box algorithm, $\text{uBBC}(\mathcal{F}_n)/\lambda$, corresponds to the parallel time if all λ evaluations are performed on parallel processors. By (2) $\text{uBBC}(\mathcal{F}_n)/\lambda$ is non-increasing with λ , and for $\lambda \leq \lambda^*$ it decreases by a factor of $\Theta(\lambda)$. Such speedups were called *linear speedups* in [11].

The $(1+\lambda)$ EA maintains the current best search point x and creates λ offspring by flipping each bit in x independently with probability p (with default $p = 1/n$). The best offspring replaces its parent if it has fitness at least $f(x)$.

3 Parallel Black-Box Complexity of LeadingOnes

We consider the function $\text{LO}(x) := \sum_{i=1}^n \prod_{j=1}^i x_j$, counting the number of leading ones in x . Similarly, $\text{LZ}(x)$ counts the number of leading zeros in x . We first provide a tool for estimating the progress made by λ trials, which may or may not be independent. It is based on moment-generating functions (mgf).

Lemma 1. *Given $X_1, \dots, X_\lambda \in \mathbb{N}$, where X_i s are random variables, not necessarily independent. Define $X_{(\lambda)} := \max_{i \in [\lambda]} X_i$, if there exists $\eta, D \geq 0$, such that for all $i \in [\lambda]$, it holds $E(e^{\eta X_i}) \leq D$, then $E(X_{(\lambda)}) \leq (\ln(D\lambda) + 1)/\eta$.*

Proof. Note first that for any $i \in [\lambda]$ and $j \in \mathbb{N}$, it follows from Markov's inequality that $\Pr(X_i \geq j) = \Pr(e^{\eta X_i} \geq e^{\eta j}) \leq e^{-\eta j} E(e^{\eta X_i}) \leq e^{-\eta j} D$. Now, let $k := \ln(D\lambda)/\eta$. It then follows by a union bound that

$$\begin{aligned} E(X_{(\lambda)}) &= \sum_{i=1}^{\infty} \Pr(X_{(\lambda)} \geq i) \leq k + \sum_{i=1}^{\infty} \Pr(X_{(\lambda)} \geq k + i) \\ &\leq k + \sum_{i=1}^{\infty} \sum_{j=1}^{\lambda} \Pr(X_j \geq k + i) \leq k + \sum_{i=1}^{\infty} \lambda e^{-\eta(k+i)} D \\ &= k + e^{-\eta k} \frac{D\lambda}{e^{\eta} - 1} \leq k + e^{-\eta k} D\lambda/\eta = (\ln(D\lambda) + 1)/\eta. \end{aligned}$$

For the LO function, the λ -parallel black-box complexity is as follows.

Theorem 1. *The λ -parallel unbiased black-box complexity of LO is*

$$\Omega\left(\frac{\lambda n}{\ln(\lambda/n)} + n^2\right) \quad \text{and} \quad O(\lambda n + n^2).$$

The cut-off point is $\lambda_{LO}^ = n$. The corresponding parallel time for an optimal algorithm is $\Omega\left(\frac{n}{\ln(\lambda/n)} + \frac{n^2}{\lambda}\right)$ and $O\left(n + \frac{n^2}{\lambda}\right)$.*

This result solves an open problem from [11], confirming that the analysis of the realm of linear speedups for LO from [11] is tight.

Proof (of Theorem 1). The upper bound follows from Lässig and Sudholt [12, Theorem 1] for a $(1+\lambda)$ EA, as within the context of this bound the $(1+\lambda)$ EA is equivalent to an island model with complete communication topology.

A lower bound $\Omega(n^2)$ follows from [13], hence the statement holds for the case $\lambda = O(n)$. In case that $\lambda = \omega(n)$, we proceed by drift analysis. Let the “potential” of a search point x be $\max_{0 \leq j \leq t, 1 \leq i \leq \lambda} \{LO(x^i(j)), LZ(x^i(j)), n/2\}$, and define the potential of the algorithm, P_t at time t to be the largest potential among all search points produced until time t .

Assume that the potential in generation t is $P_t = k$. In any generation t , let X_i for $i \in [\lambda]$ be the indicator variable for the event that all of the first $k + 1$ bit-positions in individual i are 1-bits (or 0-bits). Furthermore, let Y_i be the number of consecutive 1-bits (respectively 0-bits) from position $k + 2$ and onwards, i.e., the number of “free riders”.

Following the same arguments as in [13], the probability that $X_i = 1$ is no more than $1/(k + 1) = O(1/n)$. Defining $M := \sum_{i=1}^{\lambda} X_i$, we therefore have $E(M) = O(\lambda/n)$. Each random variable Y_i , $i \in [\lambda]$, is stochastically dominated by a geometric random variable Z_i with parameter $1/2$. The expected progress in potential is therefore

$$E(\Delta_{(\lambda)}) = E\left(\max_{i \in [\lambda]} X_i Y_i\right) \leq E\left(\max_{i \in [M]} Z_i\right).$$

The mgf of the geometric random variable Z_i is $M_{Z_i}(\eta) = 1/(2 - e^\eta)$. The tower property of the expectation and Lemma 1 with $\eta := \ln(3/2)$ and $D := 2$ give

$$\begin{aligned} E(\Delta_{(\lambda)}) &\leq E\left(E\left(\max_{i \in [M]} Z_i \mid M\right)\right) \\ &\leq E((\log(DM) + 1)/\eta) \leq (\log(E(DM)) + 1)/\eta = O(\log(\lambda/n)), \end{aligned}$$

where the last inequality follows from Jensen’s inequality and the last equality follows from $\log(\lambda/n) = \Omega(1)$. With overwhelmingly high probability, the initial potential is at least $n/2$. Hence, by classical additive drift theorems [8], the expected number of rounds to reach the optimum is $\Omega(n/\log(\lambda/n))$. Multiplying by λ gives the number of function evaluations.

4 Parallel Black-Box Complexity of Functions with Unique Optimum

De Jong, Jansen, and Wegener [9] considered the $(1+\lambda)$ EA and established a cut-off point for λ where the running time increases from $\Theta(n \log n)$ to $\omega(n \log n)$:

$$\lambda_{(1+\lambda) \text{ EA on ONEMAX}}^* = \Theta((\ln n)(\ln \ln n)/(\ln \ln \ln n)) \tag{3}$$

Recently, He, Chen, and Yao [7] presented the following tight bound for all λ :

Theorem 2 (He, Chen, Yao [7]). *The expected optimisation time of the $(1+\lambda)$ EA on ONEMAX for $\lambda \geq 3$ is*

$$\Theta\left(n \cdot \frac{\lambda \ln \ln \lambda}{\ln \lambda} + n \log n\right).$$

We show that the parallel black-box complexity is lower than the bound from Theorem 2 for large λ by a factor of $\ln \ln \lambda$.

Theorem 3. *For any $\lambda \leq e^{\sqrt{n}}$ the λ -parallel unbiased unary black-box complexity for any function with a unique optimum is at least*

$$\Omega\left(\frac{\lambda n}{\ln \lambda} + n \log n\right).$$

This bound is tight for ONEMAX, where the cut-off point is

$$\lambda_{\text{ONEMAX}}^* = \Theta(\log(n) \cdot \log \log n).$$

The corresponding parallel time for an optimal algorithm is $\Omega\left(\frac{n}{\ln \lambda} + \frac{n \log n}{\lambda}\right)$.

Note that the cut-off point is higher than the cut-off point for the $(1+\lambda)$ EA with the standard mutation rate $p = 1/n$ from (3) and [9].

For the proof we consider the progress made during a round of λ variations. Let the 0-“potential” of a search point x be $\min\{|x|_0, n/(8e)\}$, where $|x|_0$ is the number of 0-bits in x . Similarly, define the 1-“potential” of a search point x as $\min\{|x|_1, n/(8e)\}$. Let s be the minimum 0-potential among all search points queried in past rounds. Let r be the number of flipped bits during a variation, then for any search point with m number of zeros, denote the progress by $\Delta(s, m, r)$. The progress is the difference between s and the potential of the new generated point, there is no progress if this difference is negative. Let Z be the number of 0-bits that flipped to 1, then there are $r - Z$ new 0-bits that were originally 1. Therefore, the number of 0-bits in the new generated search point is $m - Z + (r - Z)$ where Z can be described by the hypergeometric distribution with parameters n, m and r . We only make progress if the number of 0-bits in the new search point is less than s . Hence the progress (decrease in potential) is

$$\Delta(s, m, r) = \max\{Z - (r - Z) + (s - m), 0\} = \max\{2Z - r + s - m, 0\}.$$

We show a tail inequality for hypergeometric variables that is more precise than Chvátal’s bound [1] and use this to derive a progress bound. A proof of the former is omitted due to space restrictions.

Lemma 2. *Let Z be a hypergeometrically distributed random variable with parameters n (number of balls), m (number of red balls), and r (number of balls drawn). If $m < n/(2e)$ then for any $z \geq r/2$, $\Pr(Z = z) \leq (2em/n)^z$.*

Lemma 3. *Let $\Delta_{(\lambda)} = \Delta_{(\lambda)}(s, m_i, r_i)$ be the maximum of λ random variables $\Delta(s, m_i, r_i)$ for arbitrary $s \leq m_i \leq n/2$ and $r_i, 1 \leq i \leq \lambda$. For $s \leq n/(8e)$ we have $E(\Delta_{(\lambda)}) = O(\log(\lambda))$.*

Proof. If $\frac{n}{4e} < m_i \leq n/2$ then we use Chvátal’s tail bound [1]: $\Pr(Z \geq E(Z) + r\delta) \leq \exp(-2\delta^2 r)$, where $E(Z) = \frac{rm}{n}$, then:

$$\begin{aligned} \Pr(\Delta(s, m_i, r_i) > 0) &= \Pr\left(Z \geq \frac{r_i + m_i - s}{2}\right) \\ &= \Pr\left(Z \geq \frac{r_i m_i}{n} + r_i \cdot \left(\frac{r_i + m_i - s}{2r_i} - \frac{m_i}{n}\right)\right) \\ &\leq \Pr\left(Z \geq E(Z) + r_i \cdot \frac{n}{8er_i}\right) \leq \exp\left(-\frac{n^2}{32e^2 r_i}\right) \end{aligned}$$

This means that the probability of making any progress is exponentially small, for any r_i . Thus we assume that $m_i \leq \frac{n}{4e}$ for all i in the following. Applying Lemma 2 to a hypergeometric random variable Z_i with parameters m_i and r_i we have, for all $z \in \mathbb{N}_0$,

$$\begin{aligned} \Pr(\Delta(s, m_i, r_i) = z) \\ = \Pr\left(Z_i = \frac{z + r_i + m_i - s}{2}\right) \leq \left(\frac{2em_i}{n}\right)^{(z+r_i+m_i-s)/2} \leq \left(\frac{1}{2}\right)^{z/2} \end{aligned}$$

hence $E(e^{\eta Z_i}) \leq D$ for $\eta := \ln(4/3)$ and $D := 9 + 6\sqrt{2}$. Applying Lemma 1 proves $E(\Delta_{(\lambda)}) = O(\log \lambda)$.

Proof (of Theorem 3). The upper bound for ONEMAX will be shown later in Theorem 4. The lower bound $\Omega(n \log n)$ follows from unbiased unary black-box complexity [13]. Hence, it suffices to prove the lower bound $\Omega(\lambda n / \ln \lambda)$.

Without loss of generality, we assume that the search point 1^n is the optimum. Following [13], we assume a “mirrored” sampling process, where every time a bit string x is queried (including in the initial generation), the algorithm queries the complement bit string \bar{x} for “free”. Hence, the 1-potential and the 0-potential (as defined above) are the same after each generation, and we apply drift analysis with respect to this potential. Variation of a search point with m 1-bits is symmetric to a variation of a search point with $n - m$ 1-bits, hence we can assume $m \leq n/2$. By a Chernoff bound, the initial potential is $n/(8e)$ with overwhelmingly high probability. Let Δ_0 be the progress due to reduction of the 0-potential, and Δ_1 be the progress due to reduction of the 1-potential. By Lemma 3, the expected change in potential per round is no more than $\max\{\Delta_0, \Delta_1\} = O(\log \lambda)$. Hence, by the additive drift theorem [8], the expected number of rounds until one of the search points 0^n or 1^n is obtained is $\Omega(n/\log \lambda)$. Multiplying by λ proves the claim.

5 An Optimal Parallel Black-Box Algorithm for OneMax

The following theorem shows that the lower bound on the black-box complexity from Theorem 3 is tight. We show that the $(1+\lambda)$ EA has a better optimisation time if the mutation rate is chosen adaptively, according to the current best fitness. This is similar to common ideas from artificial immune systems, particularly the clonal selection algorithm. Adaptive mutation rates for ONEMAX have been studied by Zarges [19], however the standard parameters for the clonal selection algorithm were too drastic to even obtain polynomial running times. Better results were obtained when using a population-based adaptation [20].

The following result reveals an optimal choice for the mutation rate of the $(1+\lambda)$ EA, depending on n and λ .

Theorem 4. *On OneMax, the expected number of function evaluations of the $(1+\lambda)$ EA with an adaptive mutation rate $p = \max\{\ln(\lambda)/(n \ln(en/i)), 1/n\}$, where i is the number of zeros in the current search point, for any $\lambda \leq e^{\sqrt{n}}$, is at most*

$$O\left(\frac{\lambda n}{\ln \lambda} + n \log n\right).$$

The parallel time (number of generations) is $O\left(\frac{n}{\ln \lambda} + \frac{n \log n}{\lambda}\right)$.

Proof. Let i be the current number of zeros and p be the mutation rate. The probability of decreasing the number of zeros by any $k \in \mathbb{N}$ with $k \leq i$ is at least

$$\begin{aligned} \Pr(\Delta \geq k) &\geq \binom{i}{k} \cdot p^k \cdot (1-p)^{n-k} \\ &\geq \frac{i^k}{k^k} \cdot p^k \cdot (1-p)^{n-k} = (1-p)^{n-k} \cdot \left(\frac{ip}{k}\right)^k. \end{aligned}$$

Then the probability that one of λ offspring will decrease the number of zeros by at least k is at least, using $1 - (1-p)^\lambda \geq 1 - e^{-p\lambda} \geq 1 - 1/(1+p\lambda) = p\lambda/(1+p\lambda)$,

$$\Pr(\Delta_{(\lambda)} \geq k) \geq 1 - (1 - \Pr(\Delta \geq k))^\lambda \geq \frac{\lambda(1-p)^{n-k} \cdot (ip/k)^k}{1 + \lambda(1-p)^{n-k} \cdot (ip/k)^k}.$$

Hence for any $k \leq i$ the expected drift is at least

$$E(\Delta_{(\lambda)}) \geq k \cdot \frac{\lambda(1-p)^{n-k} \cdot (ip/k)^k}{1 + \lambda(1-p)^{n-k} \cdot (ip/k)^k}.$$

For $i > en/\ln \lambda$, which implies $pn > 1$, we set $k := pn = \ln(\lambda)/\ln(en/i)$. We have $k \leq i$ since $k \leq \ln(\lambda) \leq \sqrt{n} \leq en/\ln \lambda$. We use $k := 1$ for $i \leq en/\ln \lambda$, the realm where $p = 1/n$. This results in the following drift function h :

$$h(i) := \begin{cases} \frac{\lambda(1-1/n)^{n-1} \cdot i/n}{1 + \lambda(1-1/n)^{n-1} \cdot i/n} & \text{if } i \leq en/\ln \lambda \\ pn \cdot \frac{\lambda(1-p)^{n-pn} \cdot (i/n)^{pn}}{1 + \lambda(1-p)^{n-pn} \cdot (i/n)^{pn}} & \text{otherwise} \end{cases}$$

We estimate the number of function evaluations by multiplying the number of generations by λ . The number of generations is estimated using Johannsen’s variable drift theorem [10] in the variant from [16], with the above function h . This gives an upper bound of

$$\begin{aligned} \frac{\lambda}{h(1)} + \int_1^n \frac{\lambda}{h(i)} \, di &= \frac{1 + \lambda(1 - 1/n)^{n-1} \cdot 1/n}{(1 - 1/n)^{n-1} \cdot 1/n} + \lambda \int_1^n \frac{1}{h(i)} \, di \\ &\leq \lambda + en + \lambda \int_1^{en/\ln \lambda} \frac{1}{h(i)} \, di + \lambda \int_{en/\ln \lambda}^n \frac{1}{h(i)} \, di. \end{aligned}$$

The first terms are at most

$$\begin{aligned} &\lambda + en + \lambda \int_1^{en/\ln \lambda} \frac{1 + \lambda(1 - 1/n)^{n-1} \cdot i/n}{\lambda(1 - 1/n)^{n-1} \cdot i/n} \, di \\ &\leq \frac{\lambda en}{\ln \lambda} + en \left(1 + \int_1^{en/\ln \lambda} \frac{1}{i} \, di \right) \leq \frac{\lambda en}{\ln \lambda} + en \cdot (2 + \ln n). \end{aligned}$$

The second integral is bounded as

$$\begin{aligned} &\int_{en/\ln \lambda}^n \frac{1 + \lambda(1 - p)^{n-pn} \cdot (i/n)^{pn}}{pn \cdot (1 - p)^{n-pn} \cdot (i/n)^{pn}} \, di \\ &\leq \int_0^n \frac{\lambda \ln(en/i)}{\ln \lambda} \, di + \frac{1}{\ln \lambda} \int_{en/\ln \lambda}^n \frac{\ln(en/i)}{e^{-pn} \cdot (i/n)^{pn}} \, di \\ &= \frac{2\lambda n}{\ln \lambda} + \frac{1}{\ln \lambda} \int_{en/\ln \lambda}^n \ln(en/i) \cdot (en/i)^{pn} \, di \\ &= \frac{2\lambda n}{\ln \lambda} + \frac{1}{\ln \lambda} \int_{en/\ln \lambda}^n \ln(en/i) \cdot \lambda \, di \leq \frac{3\lambda n}{\ln \lambda}. \end{aligned}$$

Together, we get an upper bound of $(3 + e)\lambda n / \ln(\lambda) + en \cdot (2 + \ln n)$.

6 Conclusions

We have introduced the parallel unbiased black-box complexity to quantify the limits on the performance of parallel search heuristics, including offspring populations. We proved that *every* λ -parallel unbiased black-box algorithm needs at least $\Omega(\lambda n / \log(\lambda) + n \log n)$ function evaluations on every function with unique optimum, and at least $\Omega(\lambda n / (\log(\lambda/n)) + n^2)$ function evaluations on LO. Corresponding parallel times are by a factor of λ smaller. For LO and ONEMAX we identified the cut-off point for λ , above which the asymptotic number of function evaluations increases, compared to non-parallel algorithms ($\lambda = 1$). All smaller λ allow for linear speedups with regard to the parallel time. For ONEMAX this cut-off point is higher than that for the standard $(1+\lambda)$ EA; optimal performance for all λ is achieved by a $(1+\lambda)$ EA with an adaptive mutation rate.

References

1. Chvátal, V.: The tail of the hypergeometric distribution. *Discrete Math.* 25(3), 285–287 (1979)
2. Doerr, B., Doerr, C., Ebel, F.: Lessons from the black-box: fast crossover-based genetic algorithms. In: *Proc. of GECCO 2013*, pp. 781–788. ACM (2013)
3. Doerr, B., Johannsen, D., Kötzing, T., Lehre, P.K., Wagner, M., Winzen, C.: Faster black-box algorithms through higher arity operators. In: *Proc. of FOGA 2011*, pp. 163–172. ACM (2011)
4. Doerr, B., Winzen, C.: Towards a complexity theory of randomized search heuristics: Ranking-based black-box complexity. In: Kulikov, A., Vereshchagin, N. (eds.) *CSR 2011*. LNCS, vol. 6651, pp. 15–28. Springer, Heidelberg (2011)
5. Doerr, B., Winzen, C.: Playing Mastermind with Constant-Size Memory. *Theory of Computing Systems* (2012)
6. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems* 39(4), 525–544 (2006)
7. He, J., Chen, T., Yao, X.: Average drift analysis and its application. *CoRR*, abs/1308.3080 (2013)
8. He, J., Yao, X.: A Study of Drift Analysis for Estimating Computation Time of Evolutionary Algorithms. *Natural Computing* 3(1), 21–35 (2004)
9. Jansen, T., De Jong, K.A., Wegener, I.: On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation* 13, 413–440 (2005)
10. Johannsen, D.: *Random Combinatorial Structures and Randomized Search Heuristics*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany and the Max-Planck-Institut für Informatik (2010)
11. Lässig, J., Sudholt, D.: General upper bounds on the running time of parallel evolutionary algorithms. *Evolutionary Computation* (in press), http://www.mitpressjournals.org/doi/pdf/10.1162/EVC0_a_00114
12. Lässig, J., Sudholt, D.: Analysis of speedups in parallel evolutionary algorithms for combinatorial optimization. In: Asano, T., Nakano, S.-i., Okamoto, Y., Watanabe, O. (eds.) *ISAAC 2011*. LNCS, vol. 7074, pp. 405–414. Springer, Heidelberg (2011)
13. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. *Algorithmica* 64(4), 623–642 (2012)
14. Luque, G., Alba, E.: *Parallel Genetic Algorithms—Theory and Real World Applications*. Springer (2011)
15. Mambrini, A., Sudholt, D., Yao, X.: Homogeneous and heterogeneous island models for the set cover problem. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I*. LNCS, vol. 7491, pp. 11–20. Springer, Heidelberg (2012)
16. Rowe, J.E., Sudholt, D.: The choice of the offspring population size in the $(1, \lambda)$ EA. In: *Proc. of GECCO 2012*, pp. 1349–1356 (2012)
17. Rowe, J.E., Vose, M.D.: Unbiased black box search algorithms. In: *Proc. of GECCO 2011*, p. 2035. ACM, New York (2011)
18. Teytaud, O., Gelly, S.: General lower bounds for evolutionary algorithms. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) *PPSN 2006*. LNCS, vol. 4193, pp. 21–31. Springer, Heidelberg (2006)
19. Zarges, C.: Rigorous runtime analysis of inversely fitness proportional mutation rates. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) *PPSN 2008*. LNCS, vol. 5199, pp. 112–122. Springer, Heidelberg (2008)
20. Zarges, C.: On the utility of the population size for inversely fitness proportional mutation rates. In: *Proc. of FOGA 2009*, pp. 39–46. ACM (2009)

A Generalized Markov-Chain Modelling Approach to $(1, \lambda)$ -ES Linear Optimization

Alexandre Chotard¹ and Martin Holeňa²

¹ INRIA Saclay-Ile-de-France, LRI, University Paris-Sud, France

`alexandre.chotard@lri.fr`

² Institute of Computer Science, Academy of Sciences, Pod vodárenskou věží 2,

Prague, Czech Republic

`martin@cs.cas.cz`

Abstract. Several recent publications investigated Markov-chain modelling of linear optimization by a $(1, \lambda)$ -ES, considering both unconstrained and linearly constrained optimization, and both constant and varying step size. All of them assume normality of the involved random steps, and while this is consistent with a black-box scenario, information on the function to be optimized (e.g. separability) may be exploited by the use of another distribution. The objective of our contribution is to complement previous studies realized with normal steps, and to give sufficient conditions on the distribution of the random steps for the success of a constant step-size $(1, \lambda)$ -ES on the simple problem of a linear function with a linear constraint. The decomposition of a multidimensional distribution into its marginals and the copula combining them is applied to the new distributional assumptions, particular attention being paid to distributions with Archimedean copulas.

Keywords: Evolution strategies, continuous optimization, linear optimization, linear constraint, linear function, Markov chain models, Archimedean copulas.

1 Introduction

Evolution Strategies (ES) are Derivative Free Optimization (DFO) methods, and as such are suited for the optimization of numerical problems in a black-box context, where the algorithm has no information on the function f it optimizes (e.g. existence of gradient) and can only query the function's values. In such a context, it is natural to assume normality of the random steps, as the normal distribution has maximum entropy for given mean and variance, meaning that it is the most general assumption one can make without the use of additional information on f . However such additional information may be available, and then using normal steps may not be optimal. Cases where different distributions have been studied include so-called Fast Evolution Strategies [1] or SNES [2,3] which exploits the separability of f , or heavy-tail distributions on multimodal problems [4,3].

In several recent publications [5,6,7,8], attention has been paid to Markov-chain modelling of linear optimization by a $(1, \lambda)$ -ES, i.e. by an evolution strategy

in which λ children are generated from a single parent $\mathbf{X} \in \mathbb{R}^n$ by adding normally distributed n -dimensional random steps \mathbf{M} ,

$$\mathbf{X} \leftarrow \mathbf{X} + \sigma \mathbf{C}^{\frac{1}{2}} \mathbf{M}, \text{ where } \mathbf{M} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n). \quad (1)$$

Here, σ is called step size, \mathbf{C} is a covariance matrix, and $\mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ denotes the n -dimensional standard normal distribution with zero mean and covariance matrix identity. The best among the λ children, i.e. the one with the highest fitness, becomes the parent of the next generation, and the step-size σ and the covariance matrix \mathbf{C} may then be adapted to increase the probability of sampling better children. In this paper we relax the normality assumption of the movement \mathbf{M} to a more general distribution H .

The linear function models a situation where the step-size is relatively small compared to the distance towards a local optimum. This is a simple problem that must be solved by any effective evolution strategy by diverging with positive increments of $\nabla f \cdot \mathbf{M}$. This unconstrained case was studied in [7] for normal steps with cumulative step-size adaptation (the step-size adaptation mechanism in CMA-ES [9]).

Linear constraints naturally arise in real-world problems (e.g. need for positive values, box constraints) and also model a step-size relatively small compared to the curvature of the constraint. Many techniques to handle constraints in randomised algorithms have been proposed (see [10]). In this paper we focus on the resampling method, which consists in resampling any unfeasible candidate until a feasible one is sampled. We chose this method as it makes the algorithm easier to study, and is consistent with the previous studies assuming normal steps [11,5,6,8], studying constant step-size, self adaptation and cumulative step-size adaptation mechanisms (with fixed covariance matrix).

Our aim is to study the $(1, \lambda)$ -ES with constant step-size, constant covariance matrix and random steps with a general absolutely continuous distribution H optimizing a linear function under a linear constraint handled through resampling. We want to extend the results obtained in [5,8] using the theory of Markov chains. It is our hope that such results will help in designing new algorithms using information on the objective function to make non-normal steps. We pay a special attention to distributions with Archimedean copulas, which are a particularly well transparent alternative to the normal distribution. Such distributions have been recently considered in the Estimation of Distribution Algorithms [12,13], continuing the trend of using copulas in that kind of evolutionary optimization algorithms [14].

In the next section, the basic setting for modelling the considered evolutionary optimization task is formally defined. In Section 3, the distributions of the feasible steps and of the selected steps are linked to the distribution of the random steps, and another way to sample them is provided. In Section 4, it is shown that, under some conditions on the distribution of the random steps, the normalized distance to the constraint defined in (5) is a ergodic Markov chain, and a law of large numbers for Markov chains is applied. Finally, Section 5 gives properties on the distribution of the random steps under which some of the aforementioned conditions are verified.

Due to a lack of space proofs were not included in this paper, and can instead be found at <http://hal.inria.fr/docs/01/00/30/15/PDF/ppsn2014TRlinearconstraintgeneraldistributions.pdf>.

Notations

For $(a, b) \in \mathbb{N}^2$ with $a < b$, $[a..b]$ denotes the set of integers i such that $a \leq i \leq b$. For X and Y two random vectors, $X \stackrel{(d)}{=} Y$ denotes that these variables are equal in distribution, $X \stackrel{a.s.}{\rightarrow} Y$ and $X \xrightarrow{\mathcal{P}} Y$ denote, respectively, almost sure convergence and convergence in probability. For $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n$, $\mathbf{x} \cdot \mathbf{y}$ denotes the scalar product between the vectors \mathbf{x} and \mathbf{y} , and for $i \in [1..n]$, $[\mathbf{x}]_i$ denotes the i^{th} coordinate of \mathbf{x} . For A a subset of \mathbb{R}^n , $\mathbb{1}_A$ denotes the indicator function of A . For \mathcal{X} a topological set, $\mathcal{B}(\mathcal{X})$ denotes the Borel algebra on \mathcal{X} .

2 Problem Setting and Algorithm Definition

Throughout this paper, we study a $(1, \lambda)$ -ES optimizing a linear function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ where $\lambda \geq 2$ and $n \geq 2$, with a linear constraint $g : \mathbb{R}^n \rightarrow \mathbb{R}$, handling the constraint by resampling unfeasible solutions until a feasible solution is sampled.

Take $(\mathbf{e}_k)_{k \in [1..n]}$ a orthonormal basis of \mathbb{R}^n . We may assume ∇f to be normalized as the behaviour of an ES is invariant to the composition of the objective function by a strictly increasing function (e.g. $h : x \mapsto x/\|\nabla f\|$), and the same holds for ∇g since our constraint handling method depends only on the inequality $g(\mathbf{x}) \leq 0$ which is invariant to the composition of g by a homothetic transformation. Hence w.l.o.g. we assume that $\nabla f = \mathbf{e}_1$ and $\nabla g = \cos \theta \mathbf{e}_1 + \sin \theta \mathbf{e}_2$ with the set of feasible solutions $\mathcal{X}_{\text{feasible}} := \{\mathbf{x} \in \mathbb{R}^n | g(\mathbf{x}) \leq 0\}$. We restrict our study to $\theta \in (0, \pi/2)$. Overall the problem reads

$$\begin{aligned} &\text{maximize } f(\mathbf{x}) = [\mathbf{x}]_1 \text{ subject to} \\ &g(\mathbf{x}) = [\mathbf{x}]_1 \cos \theta + [\mathbf{x}]_2 \sin \theta \leq 0 . \end{aligned} \tag{2}$$

At iteration $t \in \mathbb{N}$, from a so-called parent point $\mathbf{X}_t \in \mathcal{X}_{\text{feasible}}$ and with step-size $\sigma_t \in \mathbb{R}_+^*$ we sample new candidate solutions by adding to \mathbf{X}_t a random vector $\sigma_t \mathbf{M}_t^{i,j}$ where $\mathbf{M}_t^{i,j}$ is called a random step and $(\mathbf{M}_t^{i,j})_{i \in [1..\lambda], j \in \mathbb{N}, t \in \mathbb{N}}$ is a i.i.d. sequence of random vectors with distribution H . The i index stands for the λ new samples to be generated, and the j index stands for the unbounded number of samples used by the resampling. We denote \mathbf{M}_t^i a feasible step, that is the first element of $(\mathbf{M}_t^{i,j})_{j \in \mathbb{N}}$ such that $\mathbf{X}_t + \sigma_t \mathbf{M}_t^i \in \mathcal{X}_{\text{feasible}}$ (random steps are sampled until a suitable candidate is found). The i^{th} feasible solution \mathbf{Y}_t^i is then

$$\mathbf{Y}_t^i := \mathbf{X}_t + \sigma_t \mathbf{M}_t^i . \tag{3}$$

Then we denote $\star := \operatorname{argmax}_{i \in [1..\lambda]} f(\mathbf{Y}_t^i)$ the index of the feasible solution maximizing the function f , and update the parent point

$$\mathbf{X}_{t+1} := \mathbf{Y}_t^\star = \mathbf{X}_t + \sigma_t \mathbf{M}_t^\star , \tag{4}$$

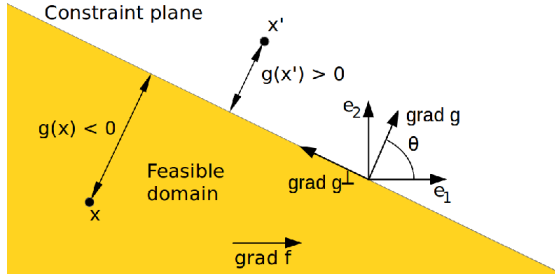


Fig. 1. Linear function with a linear constraint, in the plane spanned by ∇f and ∇g , with the angle from ∇f to ∇g equal to $\theta \in (0, \pi/2)$. The point \mathbf{x} is at distance $g(\mathbf{x})$ from the constraint hyperplan $g(\mathbf{x}) = 0$.

where \mathbf{M}_t^* is called the selected step. Then the step-size σ_t , the distribution of the random steps H or other internal parameters may be adapted.

Following [5,6,11,8] we define δ_t as

$$\delta_t := -\frac{g(\mathbf{X}_t)}{\sigma_t} . \tag{5}$$

3 Distribution of the Feasible and Selected Steps

In this section we link the distributions of the random vectors \mathbf{M}_t^i and \mathbf{M}_t^* to the distribution of the random steps $\mathbf{M}_t^{i,j}$, and give another way to sample \mathbf{M}_t^i and \mathbf{M}_t^* not requiring an unbounded number of samples.

Lemma 1. *Let a $(1, \lambda)$ -ES optimize the problem defined in (2) handling constraint through resampling. Take H the distribution of the random step $\mathbf{M}_t^{i,j}$, and for $\delta \in \mathbb{R}_+$ denote $L_\delta := \{\mathbf{x} \in \mathbb{R}^n | g(\mathbf{x}) \leq \delta\}$. Providing that H is absolutely continuous and that $H(L_\delta) > 0$ for all $\delta \in \mathbb{R}_+$, the distribution \tilde{H}_δ of the feasible step and \tilde{H}_δ^* the distribution of the selected step when $\delta_t = \delta$ are absolutely continuous, and denoting h, \tilde{h}_δ and \tilde{h}_δ^* the probability density functions of, respectively, the random step, the feasible step \mathbf{M}_t^i and the selected step \mathbf{M}_t^* when $\delta_t = \delta$*

$$\tilde{h}_\delta(\mathbf{x}) = \frac{h(\mathbf{x}) \mathbb{1}_{L_\delta}(\mathbf{x})}{H(L_\delta)} , \tag{6}$$

and

$$\begin{aligned} \tilde{h}_\delta^*(\mathbf{x}) &= \lambda \tilde{h}_\delta(\mathbf{x}) \tilde{H}_\delta((-\infty, [\mathbf{x}]_1) \times \mathbb{R}^{n-1})^{\lambda-1} \\ &= \lambda \frac{h(\mathbf{x}) \mathbb{1}_{L_\delta}(\mathbf{x}) H((-\infty, [\mathbf{x}]_1) \times \mathbb{R}^{n-1} \cap L_\delta)^{\lambda-1}}{H(L_\delta)^\lambda} . \end{aligned} \tag{7}$$

The vectors $(\mathbf{M}_t^i)_{i \in [1.. \lambda]}$ and \mathbf{M}_t^* are functions of the vectors $(\mathbf{M}_t^{i,j})_{i \in [1.. \lambda], j \in \mathbb{N}}$ and of δ_t . In the following Lemma an equivalent way to sample \mathbf{M}_t^i and \mathbf{M}_t^*

is given which uses a finite number of samples. This method is useful if one wants to avoid dealing with the infinite dimension space implied by the sequence $(\mathbf{M}_t^{i,j})_{i \in [1.. \lambda], j \in \mathbb{N}}$.

Lemma 2. *Let a $(1, \lambda)$ -ES optimize problem (2), handling the constraint through resampling, and take δ_t as defined in (5). Let H denote the distribution of $\mathbf{M}_t^{i,j}$ that we assume absolutely continuous, $\nabla g^\perp := -\sin \theta \mathbf{e}_1 + \cos \theta \mathbf{e}_2$, \mathbf{Q} the rotation matrix of angle θ changing $(\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n)$ into $(\nabla g, \nabla g^\perp, \dots, \mathbf{e}_n)$. Take $F_{1,\delta}(x) := \Pr(\mathbf{M}_t^i \cdot \nabla g \leq x | \delta_t = \delta)$, $F_{2,\delta}(x) := \Pr(\mathbf{M}_t^i \cdot \nabla g^\perp \leq x | \delta_t = \delta)$ and $F_{k,\delta}(x) := \Pr([\mathbf{M}_t^i]_k \leq x | \delta_t = \delta)$ for $k \in [3..n]$, the marginal cumulative distribution functions when $\delta_t = \delta$, and C_δ the copula of $(\mathbf{M}_t^i \cdot \nabla g, \mathbf{M}_t^i \cdot \nabla g^\perp, \dots, \mathbf{M}_t^i \cdot \mathbf{e}_n)$. We define*

$$\mathcal{G} : (\delta, (u_i)_{i \in [1..n]}) \in \mathbb{R}_+ \times [0, 1]^n \mapsto \mathbf{Q} \begin{pmatrix} F_{1,\delta}^{-1}(u_1) \\ \vdots \\ F_{n,\delta}^{-1}(u_n) \end{pmatrix}, \tag{8}$$

$$\mathcal{G}^* : (\delta, (\mathbf{v}_i)_{i \in [1.. \lambda]}) \in \mathbb{R}_+ \times [0, 1]^{n\lambda} \mapsto \operatorname{argmax}_{\mathbf{G} \in \{\mathcal{G}(\delta, \mathbf{v}_i) | i \in [1.. \lambda]\}} f(\mathbf{G}). \tag{9}$$

Then, if the copula C_δ is constant in regard to δ , for $\mathbf{W}_t = (\mathbf{V}_{i,t})_{i \in [1.. \lambda]}$ a i.i.d. sequence with $\mathbf{V}_{i,t} \sim C_\delta$

$$\mathcal{G}(\delta_t, \mathbf{V}_{i,t}) \stackrel{(d)}{=} \mathbf{M}_t^i, \tag{10}$$

$$\mathcal{G}^*(\delta_t, \mathbf{W}_t) \stackrel{(d)}{=} \mathbf{M}_t^*. \tag{11}$$

We may now use these results to show the divergence of the algorithm when the step-size is constant, using the theory of Markov chains [15].

4 Divergence of the $(1, \lambda)$ -ES with Constant Step-Size

Following the first part of [8], we restrict our attention to the constant step size in the remainder of the paper, that is for all $t \in \mathbb{N}$ we take $\sigma_t = \sigma \in \mathbb{R}_+^*$.

From Eq. (4), by recurrence and dividing by t , we see that

$$\frac{[\mathbf{X}_t - \mathbf{X}_0]_1}{t} = \frac{\sigma}{t} \sum_{i=0}^{t-1} \mathbf{M}_i^*. \tag{12}$$

The latter term suggests the use of a Law of Large Numbers to show the convergence of the left hand side to a constant that we call the divergence rate. The random vectors $(\mathbf{M}_t^*)_{t \in \mathbb{N}}$ are not i.i.d. so in order to apply a Law of Large Numbers on the right hand side of the previous equation we use Markov chain theory, more precisely the fact that $(\mathbf{M}_t^*)_{t \in \mathbb{N}}$ is a function of a $(\delta_t, (\mathbf{M}_t^{i,j})_{i \in [1.. \lambda], j \in \mathbb{N}})_{t \in \mathbb{N}}$ which is a geometrically ergodic Markov chain. As $(\mathbf{M}_t^{i,j})_{i \in [1.. \lambda], j \in \mathbb{N}, t \in \mathbb{N}}$ is a i.i.d. sequence, it is a Markov chain, and the sequence $(\delta_t)_{t \in \mathbb{N}}$ is also a Markov chain as stated in the following proposition.

Proposition 1. *Let a $(1, \lambda)$ -ES with constant step-size optimize problem (2), handling the constraint through resampling, and take δ_t as defined in (5). Then no matter what distribution the i.i.d. sequence $(\mathbf{M}_t^{i,j})_{i \in [1..\lambda], (j,t) \in \mathbb{N}^2}$ have, $(\delta_t)_{t \in \mathbb{N}}$ is a homogeneous Markov chain and*

$$\delta_{t+1} = \delta_t - g(\mathbf{M}_t^*) = \delta_t - \cos \theta[\mathbf{M}_t^*]_1 - \sin \theta[\mathbf{M}_t^*]_2 . \tag{13}$$

We now show ergodicity of the Markov chain $(\delta_t)_{t \in \mathbb{N}}$, which implies that the t -steps transition kernel (the function $A \mapsto \Pr(\delta_t \in A | \delta_0 = \delta)$ for $A \in \mathcal{B}(\mathbb{R}_+)$) converges towards a stationary measure π , generalizing Propositions 3 and 4 of [8].

Proposition 2. *Let a $(1, \lambda)$ -ES with constant step-size optimize problem (2), handling the constraint through resampling. We assume that the distribution of $\mathbf{M}_t^{i,j}$ is absolutely continuous with probability density function h , and that h is continuous and strictly positive on \mathbb{R}^n . Denote μ_+ the Lebesgue measure on $(\mathbb{R}_+, \mathcal{B}(\mathbb{R}_+))$, and for $\alpha > 0$ take the functions $V : \delta \mapsto \delta$, $V_\alpha : \delta \mapsto \exp(\alpha\delta)$ and $r_1 : \delta \mapsto 1$. Then $(\delta_t)_{t \in \mathbb{N}}$ is μ_+ -irreducible, aperiodic and compact sets are small sets for the Markov chain.*

If the following two additional conditions are fulfilled

$$\mathbf{E}(|g(\mathbf{M}_t^{i,j})| \mid \delta_t = \delta) < \infty \text{ for all } \delta \in \mathbb{R}_+ , \text{ and} \tag{14}$$

$$\lim_{\delta \rightarrow +\infty} \mathbf{E}(g(\mathbf{M}_t^*) | \delta_t = \delta) \in \mathbb{R}_+^* , \tag{15}$$

then $(\delta_t)_{t \in \mathbb{N}}$ is r_1 -ergodic and positive Harris recurrent with some invariant measure π .

Furthermore, if

$$\mathbf{E}(\exp(g(\mathbf{M}_t^{i,j})) | \delta_t = \delta) < \infty \text{ for all } \delta \in \mathbb{R}_+ , \tag{16}$$

then for $\alpha > 0$ small enough, $(\delta_t)_{t \in \mathbb{N}}$ is also V_α -geometrically ergodic.

We now use a law of large numbers ([15] Theorem 17.0.1) on the Markov chain $(\delta_t, (\mathbf{M}_t^{i,j})_{i \in [1..\lambda], j \in \mathbb{N}})_{t \in \mathbb{N}}$ to obtain an almost sure divergence of the algorithm.

Proposition 3. *Let a $(1, \lambda)$ -ES optimize problem (2), handling the constraint through resampling. Assume that the distribution H of the random step $\mathbf{M}_t^{i,j}$ is absolutely continuous with continuous and strictly positive density h , that conditions (16) and (15) of Proposition 2 hold, and denote π and μ_M the stationary distribution of respectively $(\delta_t)_{t \in \mathbb{N}}$ and $(\mathbf{M}_t^{i,j})_{i \in [1..\lambda], (j,t) \in \mathbb{N}^2}$. Then*

$$\frac{[\mathbf{X}_t - \mathbf{X}_0]_1}{t} \xrightarrow[t \rightarrow +\infty]{a.s.} \sigma \mathbf{E}_{\pi \times \mu_M}([\mathbf{M}_t^*]_1) . \tag{17}$$

Furthermore if $\mathbf{E}([\mathbf{M}_t^]_2) < 0$, then the right hand side of Eq. (17) is strictly positive.*

5 Application to More Specific Distributions

Throughout this section we give cases where the assumptions on the distribution of the random steps H used in Proposition 2 or Proposition 3 are verified.

The following lemma shows an equivalence between a non-identity covariance matrix for H and a different norm and constraint angle θ .

Lemma 3. *Let a $(1, \lambda)$ -ES optimize problem (2), handling the constraint with resampling. Assume that the distribution H of the random step $\mathbf{M}_t^{i,j}$ has positive definite covariance matrix \mathbf{C} with eigenvalues $(\alpha_i^2)_{i \in [1..n]}$ and take $\mathbf{B} = (b_{i,j})_{(i,j) \in [1..n]^2}$ such that \mathbf{BCB}^{-1} is diagonal. Denote $\mathcal{A}_{H,g,\mathbf{X}_0}$ the sequence of parent points $(\mathbf{X}_t)_{t \in \mathbb{N}}$ of the algorithm with distribution H for the random steps $\mathbf{M}_t^{i,j}$, constraint angle θ and initial parent \mathbf{X}_0 . Then for all $k \in [1..n]$*

$$\beta_k [\mathcal{A}_{H,\theta,\mathbf{X}_0}]_k \stackrel{(d)}{=} [\mathcal{A}_{\mathbf{C}^{-1/2}H,\theta',\mathbf{X}'_0}]_k, \tag{18}$$

where $\beta_k = \sqrt{\sum_{j=1}^n \frac{b_{j,i}^2}{\alpha_j^2}}$, $\theta' = \arccos(\frac{\beta_1 \cos \theta}{\beta_g})$ with $\beta_g = \sqrt{\beta_1^2 \cos^2 \theta + \beta_2^2 \sin^2 \theta}$, and $[\mathbf{X}'_0]_k = \beta_k [\mathbf{X}_0]_k$ for all $k \in [1..n]$.

Although Eq. (17) shows divergence of the algorithm, it is important that it diverges in the right direction, i.e. that the right hand side of Eq. (17) has a positive sign. This is achieved when the distribution of the random steps is isotropic, as stated in the following proposition.

Proposition 4. *Let a $(1, \lambda)$ -ES optimize problem (2) with constant step-size, handling the constraint with resampling. Suppose that the Markov chain $(\delta_t)_{t \in \mathbb{N}}$ is positive Harris, that the distribution H of the random step $\mathbf{M}_t^{i,j}$ is absolutely continuous with strictly positive density h , and take \mathbf{C} its covariance matrix. If the distribution $\mathbf{C}^{-1/2}H$ is isotropic then $\mathbf{E}_{\pi \times \mu_M}([\mathbf{M}_t^*]_1) > 0$.*

Lemma 3 and Proposition 4 imply the following result to hold for multivariate normal distributions.

Proposition 5. *Let a $(1, \lambda)$ -ES optimize problem (2) with constant step-size, handling the constraint with resampling. If H is a multivariate normal distribution with mean $\mathbf{0}$, then $(\delta_t)_{t \in \mathbb{N}}$ is a geometrically ergodic positive Harris Markov chain, Eq. (17) holds and its right hand side is strictly positive.*

To obtain sufficient conditions for the density of the random steps to be strictly positive, it is advantageous to decompose that distribution into its marginals and the copula combining them. We pay a particular attention to *Archimedean copulas*, i.e., copulas defined

$$(\forall \mathbf{u} \in [0, 1]^n) C_\psi(\mathbf{u}) = \psi(\psi^{-1}([\mathbf{u}]_1) + \dots + \psi^{-1}([\mathbf{u}]_n)), \tag{19}$$

where $\psi : [0, +\infty] \rightarrow [0, 1]$ is an Archimedean generator, i.e., $\psi(0) = 1, \psi(+\infty) = \lim_{t \rightarrow +\infty} \psi(t) = 0$, ψ is continuous and strictly decreasing on $[0, \inf\{t : \psi(t) = 0\})$, and ψ^{-1} denotes the generalized inverse of ψ ,

$$(\forall u \in [0, 1]) \psi^{-1}(u) = \inf\{t \in [0, +\infty] : \psi(t) = u\}. \tag{20}$$

The reason for our interest is that Archimedean copulas are invariant with respect to permutations of variables, i.e.,

$$(\forall \mathbf{u} \in [0, 1]^n) C_\psi(\mathbf{Q}\mathbf{u}) = C_\psi(\mathbf{u}). \tag{21}$$

holds for any permutation matrix $\mathbf{Q} \in \mathbb{R}^{n,n}$. This can be seen as a weak form of isotropy because in the case of isotropy, (19) holds for any rotation matrix, and a permutation matrix is a specific rotation matrix.

Proposition 6. *Let H be the distribution of the two first dimensions of the random step $\mathbf{M}_t^{i,j}$, H_1 and H_2 be its marginals, and C be the copula relating H to H_1 and H_2 . Then the following holds:*

1. *Sufficient for H to have a continuous strictly positive density is the simultaneous validity of the following two conditions.*

(i) *H_1 and H_2 have continuous strictly positive densities h_1 and h_2 , respectively.*

(ii) *C has a continuous strictly positive density c .*

Moreover, if (i) and (ii) are valid, then

$$(\forall \mathbf{x} \in \mathbb{R}^2) h(\mathbf{x}) = c(H_1([\mathbf{x}]_1), H_2([\mathbf{x}]_2))h_1([\mathbf{x}]_1)h_2([\mathbf{x}]_2). \tag{22}$$

2. *If C is Archimedean with generator ψ , then it is sufficient to replace (ii) with (ii') ψ is at least 4-monotone, i.e., ψ is continuous on $[0, +\infty]$, ψ'' is decreasing and convex on \mathbb{R}_+ , and $(\forall t \in \mathbb{R}_+) (-1)^k \psi^{(k)}(t) \geq 0, k = 0, 1, 2$.*

In this case, if (i) and (ii') are valid, then

$$(\forall \mathbf{x} \in \mathbb{R}^2) h(\mathbf{x}) = \frac{\psi''(\psi^{-1}(H_1([\mathbf{x}]_1)) + \psi^{-1}(H_2([\mathbf{x}]_2)))}{\psi'(\psi^{-1}(H_1([\mathbf{x}]_1)) + \psi^{-1}(H_2([\mathbf{x}]_2)))} h_1([\mathbf{x}]_1)h_2([\mathbf{x}]_2). \tag{23}$$

6 Discussion

The paper presents a generalization of recent results of the first author [8] concerning linear optimization by a $(1, \lambda)$ -ES in the constant step size case. The generalization consists in replacing the assumption of normality of random steps involved in the evolution strategy by substantially more general distributional assumptions. This generalization shows that isotropic distributions solve the linear problem. Also, although the conditions for the ergodicity of the studied Markov chain accept some heavy-tail distributions, an exponentially vanishing tail allow for geometric ergodicity, which imply a faster convergence to its stationary distribution, and faster convergence of Monte Carlo simulations. In our

opinion, these conditions increase the insight into the role that different kinds of distributions play in evolutionary computation, and enlarges the spectrum of possibilities for designing evolutionary algorithms with solid theoretical fundamentals. At the same time, applying the decomposition of a multidimensional distribution into its marginals and the copula combining them, the paper attempts to bring a small contribution to the research into applicability of copulas in evolutionary computation, complementing the more common application of copulas to the Estimation of Distribution Algorithms [12,14,13].

Needless to say, more realistic than the constant step size case, but also more difficult to investigate, is the varying step size case. The most important results in [8] actually concern that case. A generalization of those results for non-Gaussian distributions of random steps for cumulative step-size adaptation ([9]) is especially difficult as the evolution path is tailored for Gaussian steps, and some careful tweaking would have to be applied. The σ self-adaptation evolution strategy ([16]), studied in [6] for the same problem, appears easier, and would be our direction for future research.

Acknowledgment. The research reported in this paper has been supported by grant ANR-2010-COSI-002 (SIMINOLE) of the French National Research Agency, and Czech Science Foundation (GAČR) grant 13-17187S.

References

1. Yao, X., Liu, Y.: Fast evolution strategies. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 149–161. Springer, Heidelberg (1997)
2. Schaul, T.: Benchmarking Separable Natural Evolution Strategies on the Noiseless and Noisy Black-box Optimization Testbeds. In: Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference, Philadelphia, PA (2012)
3. Schaul, T., Glasmachers, T., Schmidhuber, J.: High dimensions and heavy tails for natural evolution strategies. In: Genetic and Evolutionary Computation Conference (GECCO) (2011)
4. Hansen, N., Gemperle, F., Auger, A., Koumoutsakos, P.: When do heavy-tail distributions help? In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 62–71. Springer, Heidelberg (2006)
5. Arnold, D.: On the behaviour of the $(1,\lambda)$ -ES for a simple constrained problem. In: Foundations of Genetic Algorithms - FOGA 2011, pp. 15–24. ACM (2011)
6. Arnold, D.V.: On the behaviour of the $(1,\lambda)$ - σ SA-ES for a constrained linear problem. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 82–91. Springer, Heidelberg (2012)
7. Chotard, A., Auger, A., Hansen, N.: Cumulative step-size adaptation on linear functions. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part I. LNCS, vol. 7491, pp. 72–81. Springer, Heidelberg (2012)

8. Chotard, A., Auger, A., Hansen, N.: Markov chain analysis of evolution strategies on a linear constraint optimization problem. In: 2014 IEEE Congress on Evolutionary Computation (CEC) (2014)
9. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2), 159–195 (2001)
10. Coello Coello, C.A.: Constraint-handling techniques used with evolutionary algorithms. In: Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation, GECCO 2008, pp. 2445–2466. ACM, New York (2008)
11. Arnold, D.V., Brauer, D.: On the behaviour of the (1+1)-ES for a simple constrained problem. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 1–10. Springer, Heidelberg (2008)
12. Cuesta-Infante, A., Santana, R., Hidalgo, J., Bielza, C., Larrañaga, P.: Bivariate empirical and n-variate archimedean copulas in estimation of distribution algorithms. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
13. Wang, L., Guo, X., Zeng, J., Hong, Y.: Copula estimation of distribution algorithms based on exchangeable archimedean copula. *International Journal of Computer Applications in Technology* 43, 13–20 (2012)
14. Salinas-Gutiérrez, R., Hernández-Aguirre, A., Villa-Diharce, E.R.: Using copulas in estimation of distribution algorithms. In: Aguirre, A.H., Borja, R.M., Garciá, C.A.R. (eds.) MICAI 2009. LNCS, vol. 5845, pp. 658–668. Springer, Heidelberg (2009)
15. Meyn, S.P., Tweedie, R.L.: Markov chains and stochastic stability, 2nd edn. Cambridge University Press (1993)
16. Beyer, H.-G.: Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation* 3(3), 311–347 (1995)

Level-Based Analysis of Genetic Algorithms and Other Search Processes

Dogan Corus¹, Duc-Cuong Dang¹, Anton V. Eremeev², and Per Kristian Lehre¹

¹ University of Nottingham, United Kingdom

² Omsk Branch of Sobolev Institute of Mathematics, Russia

Abstract. The fitness-level technique is a simple and old way to derive upper bounds for the expected runtime of simple *elitist* evolutionary algorithms (EAs). Recently, the technique has been adapted to deduce the runtime of algorithms with *non-elitist* populations and *unary* variation operators [2,8]. In this paper, we show that the restriction to unary variation operators can be removed. This gives rise to a much more general analytical tool which is applicable to a wide range of search processes. As introductory examples, we provide simple runtime analyses of many variants of the Genetic Algorithm on well-known benchmark functions, such as ONEMAX, LEADINGONES, and the sorting problem.

1 Introduction

The theoretical understanding of Evolutionary Algorithms (EAs) has advanced significantly. A contributing factor for this success may have been the strategy to analyse simple settings before proceeding to more complex scenarios, while at the same time developing appropriate analytical techniques.

The fitness-level technique is one of the oldest techniques for deriving upper bounds on the expected runtime of EAs. In this technique, the solution space is partitioned into disjoint subsets called *fitness-levels* according to ascending values of the fitness function. The expected runtime can be deduced from bounds on the probabilities of escaping the fitness levels. Applications of the technique is widely known in the literature for classical *elitist* EAs [18]. Eremeev used a fitness-level technique to obtain bounds on the expected proportion of the population of a *non-elitist* EA above a certain fitness level [5]. By generalising results in [11], the first adaptation of the fitness-level technique to run-time analysis of non-elitist population-based EAs was made in [8], and refined in [2]. One limitation of the approaches in [2,8] is that the partition must be fitness-based and only *unary variation operators* are allowed, e.g. Genetic Algorithms (GAs) are excluded. Runtime analysis of GAs has been subjected to increasing interest in the recent years (e.g. see [3,10,12,13,15]).

We show that the above limitations can be removed from [2]. This gives rise to a much more general tool which is applicable to a wide range of search processes involving non-elitist populations. As introductory examples, we analyse the runtime of variants of the Genetic Algorithm (GA) with different selection mechanisms and crossover operators on well-known functions, such as ONEMAX and LEADINGONES, and on the sorting problem.

2 Algorithmic Scheme

We consider population-based algorithms at a very abstract level in which fitness evaluations, selection and variation operations, which depending on the current population P of size λ , are represented by a distribution $D(P)$ over a finite set \mathcal{X} . More precisely, D is a mapping from \mathcal{X}^λ into the space of probability distributions over \mathcal{X} . The next generation is obtained by sampling each new individual independently from $D(P)$. This scheme is summarised below.

Algorithm 1. Population-based algorithm with independent sampling

Require:

Finite state space \mathcal{X} , and population size $\lambda \in \mathbb{N}$,
 Mapping D from \mathcal{X}^λ to the space of probability distributions over \mathcal{X} .

1. $P_0 \sim \text{Unif}(\mathcal{X}^\lambda)$
 2. **for** $t = 0, 1, 2, \dots$ until termination condition met **do**
 3. Sample $P_{t+1}(i) \sim D(P_t)$ independently for each $i \in [\lambda]$
 4. **end for**
-

A similar scheme was studied in [17], where it was called *Random Heuristic Search* with an *admissible transition rule* (see [17]). Some examples of such algorithms are Simulated Annealing (more generally any algorithm with the population composed of a single individual), Stochastic Beam Search [17], Estimation of Distribution Algorithms such as the Univariate Marginal Distribution Algorithm [1] and the Genetic Algorithm (GA) [6]. The previous studies of the framework were often limited to some restricted settings [12] or mainly focused on infinite populations [17]. In this paper, we are interested in finite populations and develop a general method to deduce the expected runtime of the search processes defined in terms of *number of evaluations*. We illustrate our methods with runtime analysis of GAs under various settings (which are different to [12]).

The term Genetic Algorithm is often applied to EAs that use recombination operators. The GA is Algorithm 1 where the sampling $y \sim D(P_t)$ is the following: (i) $u \sim p_{\text{sel}}(P_t)$, $v \sim p_{\text{sel}}(P_t)$ (selection); (ii) $\{x', x''\} \sim p_{\text{xor}}(u, v)$, $x \sim \text{Unif}(\{x', x''\})$ (crossover); (iii) $y \sim p_{\text{mut}}(x)$ (mutation). The additional part $x \sim \text{Unif}(\{x_1, x_2\})$ at crossover is to match Algorithm 1 that produces only one resulting bitstring. We call this operator the *one-offspring* version of the *standard* crossover. In the rest of this paper, the two operations of the one-offspring version will be denoted simply by $x \sim p_{\text{xor}}(x, y)$. Here the standard operators of GA are formally represented by transition matrices:

- $p_{\text{sel}} : [\lambda] \rightarrow [0, 1]$ represents a selection operator, where $p_{\text{sel}}(i|P_t)$ is the probability of selecting the i -th individual from population P_t .
- $p_{\text{mut}} : \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$, where $p_{\text{mut}}(y|x)$ is the probability of mutating $x \in \mathcal{X}$ into $y \in \mathcal{X}$.
- $p_{\text{xor}} : \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$, where $p_{\text{xor}}(x|u, v)$ is the probability of obtaining x as a result of crossover (or recombination) between $u, v \in \mathcal{X}$

3 Main Theorem

This section states a general technique for obtaining upper bounds on the expected runtime of any process that can be described in the form of Algorithm 1. We use the following notation. For any positive integer n , define $[n] := \{1, 2, \dots, n\}$. The natural logarithm is denoted by $\ln(\cdot)$. The complement of an event \mathcal{E} is denoted by $\bar{\mathcal{E}}$. Suppose that for some m there is an ordered partition of \mathcal{X} into subsets (A_1, \dots, A_{m+1}) called *levels*. For $j \in [m]$ we denote by $A_j^+ := \cup_{i=j+1}^{m+1} A_i$, the union of all levels above level j . An example of partition is the *canonical* partition, where each level regroups solutions having the same fitness value (see e.g. [8]). This partition is classified as *fitness-based*, our main theorem is not limited to this particular type of partition.

Lemma 1 (Lemma 5 and 6 in [2]). *Let $X \sim \text{Bin}(\lambda, p)$ with $p \geq (i/\lambda)(1 + \delta)$, it holds that $E[e^{-\kappa X}] \leq e^{-\kappa i}$ for any $\kappa \in (0, \delta)$. For $i \geq 1$, it also holds that $E[\ln((1 + cX)/(1 + ci))] \geq c\varepsilon$ where $\varepsilon = \min\{1/2, \delta/2\}$ and $c = \varepsilon^4/24$.*

Theorem 1. *Given a partition (A_1, \dots, A_{m+1}) of \mathcal{X} , define $T := \min\{t\lambda \mid |P_t \cap A_{m+1}| > 0\}$ to be the first point in time that elements of A_{m+1} appear in P_t of Algorithm 1. If there exist parameters $z_1, \dots, z_m, z_* \in (0, 1]$, $\delta > 0$, a constant $\gamma_0 \in (0, 1)$ and a function $z_0 : (0, \gamma_0) \rightarrow \mathbb{R}$ such that for all $j \in [m]$, $P \in \mathcal{X}^\lambda$, $y \sim D(P)$ and $\gamma \in (0, \gamma_0)$ we have*

- (G1) $\Pr(y \in A_j^+ \mid |P \cap A_{j-1}^+| \geq \gamma_0\lambda) \geq z_j \geq z_*$
- (G2) $\Pr(y \in A_j^+ \mid |P \cap A_{j-1}^+| \geq \gamma_0\lambda, |P \cap A_j^+| \geq \gamma\lambda) \geq z_0(\gamma) \geq (1 + \delta)\gamma$
- (G3) $\lambda \geq \frac{2}{a} \ln\left(\frac{16m}{ac\varepsilon z_*}\right)$ with $a = \frac{\delta^2\gamma_0}{2(1 + \delta)}$, $\varepsilon = \min\{\delta/2, 1/2\}$ and $c = \varepsilon^4/24$

then $E[T] \leq \frac{2}{c\varepsilon} \left(m\lambda(1 + \ln(1 + c\lambda)) + \sum_{j=1}^m \frac{1}{z_j} \right)$

Informally, the two first conditions require a relationship between P and the distribution $D(P)$: (G1) demands a certain probability z_j of creating an individual at level $j + 1$ when some fixed portion of the population is already at level j (or higher); (G2) requires that in the fixed portion, the number of individuals at levels strictly higher than j (if those exist) tends to increase, e.g. by a multiplicative factor of $1 + \delta$. Finally, (G3) requires a sufficiently large population size. The proof follows the same ideas as those in [2].

Proof. We use the following notation. The number of individuals in $A_j \cup A_j^+$ at generation t is denoted by X_t^j . The current level of the population at generation t is denoted by Z_t , where $Z_t := \ell$ iff $X_t^\ell \geq \lceil \gamma_0\lambda \rceil$ and $X_t^{\ell+1} < \gamma_0\lambda$. Note that Z_t is uniquely defined, as it is the level of the γ_0 -ranked individual at generation t . We also use q_j to denote the probability to generate at least one individual at a level strictly greater than j in the next generation, knowing that there are at least $\lceil \gamma_0\lambda \rceil$ individuals of the population at level j or higher in the current generation. Because of (G1), we have $q_j \geq 1 - (1 - z_j)^\lambda \geq z_j\lambda/(z_j\lambda + 1)$.

The theorem can now be proved using the additive drift theorem with respect to the potential function $g(t) := g_1(t) + g_2(t)$, where

$$g_1(t) := (m - Z_t) \ln(1 + c\lambda) - \ln(1 + cX_t^{Z_t+1})$$

$$\text{and } g_2(t) := \frac{1}{q_{Z_t} e^{\kappa X_t^{Z_t+1}}} + \sum_{j=Z_t+1}^{m-1} \frac{1}{q_j} \text{ with } \kappa \in (0, \delta)$$

The above components originated from the drift analysis of [8], then later improved by [2]. Function g is bounded from above by, $g(t) \leq m \ln(1 + c\lambda) + \sum_{j=1}^m \frac{1}{q_j} \leq m(1 + \ln(1 + c\lambda)) + \frac{1}{\lambda} \sum_{j=1}^m \frac{1}{z_j}$. At generation t , we use $R = Z_{t+1} - Z_t$ to denote the random variable describing the next progress in terms of levels. To simplify further writing, let us put $\ell = Z_t$, $i = X_t^\ell$, $X = X_{t+1}^\ell$, then $\Delta = g(t) - g(t + 1) = \Delta_1 + \Delta_2$ with

$$\Delta_1 := g_1(t) - g_1(t + 1) = R \ln(1 + c\lambda) + \ln \left(\frac{1 + X_{t+1}^{\ell+R+1}}{1 + ci} \right)$$

$$\Delta_2 := g_2(t) - g_2(t + 1) = \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_{\ell+R} e^{\kappa X_{t+1}^{\ell+R+1}}} + \sum_{j=\ell+1}^{\ell+R} \frac{1}{q_j}$$

Let us denote by \mathcal{E}_t the event that the population in the next generation does not fall down to a lower level, $\mathcal{E}_t : Z_{t+1} \geq Z_t$. We first compute the *conditional forward drift* $E[\Delta | \mathcal{F}_t, \mathcal{E}_t]$, here \mathcal{F}_t is the filtration induced by P_t . Under \mathcal{E}_t , R is a non-negative random variable and Δ is a random variable indexed by R , noted as $\Delta = Y_R$. We can show that $Y_{r \geq 1} \geq Y_0$ for fixed indexes.

$$Y_0 = \ln \left(\frac{1 + cX}{1 + ci} \right) + \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_\ell e^{\kappa X}} \leq \ln \left(\frac{1 + c\lambda}{1 + ci} \right) + \frac{1}{q_\ell e^{\kappa i}}$$

$$Y_{r \geq 1} = r \ln(1 + c\lambda) + \ln \left(\frac{1 + X_{t+1}^{\ell+r+1}}{1 + ci} \right) + \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_{\ell+r} e^{\kappa X_{t+1}^{\ell+r+1}}} + \sum_{j=\ell+1}^{\ell+r} \frac{1}{q_j}$$

$$\geq \ln(1 + c\lambda) + \ln \left(\frac{1}{1 + ci} \right) + \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_{\ell+r}} + \sum_{j=\ell+1}^{\ell+r} \frac{1}{q_j}$$

$$= \ln \left(\frac{1 + c\lambda}{1 + ci} \right) + \frac{1}{q_\ell e^{\kappa i}} + \sum_{j=\ell+1}^{\ell+r-1} \frac{1}{q_j} \geq \ln \left(\frac{1 + c\lambda}{1 + ci} \right) + \frac{1}{q_\ell e^{\kappa i}} \geq Y_0$$

It is then clear (or see Lemma 7 in [2]) that $E[\Delta | \mathcal{F}_t, \mathcal{E}_t] = E[Y_R | \mathcal{F}_t, \mathcal{E}_t] \geq E[Y_0 | \mathcal{F}_t, \mathcal{E}_t]$, so we only focus on $r = 0$ to lower bound the drift. We separate two cases, $i = 0$, the event is denoted by \mathcal{Z}_t , and $i \geq 1$ (event $\bar{\mathcal{Z}}_t$). Recall that each individual is generated independently from each other, so during $\bar{\mathcal{Z}}_t$ we have that $X \sim \text{Bin}(\lambda, p)$ where $p \geq z_0(i/\lambda)$. From (G2), we also get $z_0(i/\lambda) \geq (i/\lambda)(1 + \delta)$. Hence $p \geq (i/\lambda)(1 + \delta)$ and by Lemma 1, it holds for $i \geq 1$ (event $\bar{\mathcal{Z}}_t$) that

$$E[\Delta_1 | \mathcal{F}_t, \mathcal{E}_t, \bar{\mathcal{Z}}_t] \geq E \left[\ln \left(\frac{1 + cX}{1 + ci} \right) \middle| \mathcal{F}_t, \mathcal{E}_t, \bar{\mathcal{Z}}_t \right] \geq c\epsilon$$

$$E [\Delta_2 | \mathcal{F}_t, \mathcal{E}_t, \bar{Z}_t] \geq \frac{1}{q_\ell} (e^{-\kappa i} - E [e^{-\kappa X} | \mathcal{F}_t, \mathcal{E}_t, \bar{Z}_t]) \geq 0$$

For $i = 0$, we get $E [\Delta_1 | \mathcal{F}_t, \mathcal{E}_t, Z_t] \geq E [\ln(1) | \mathcal{F}_t, \mathcal{E}_t, Z_t] = 0$ because $X \geq 0$. Recall that $q_\ell = \Pr(X \geq 1 | \mathcal{F}_t, \mathcal{E}_t, Z_t)$, so

$$\begin{aligned} E [\Delta_2 | \mathcal{F}_t, \mathcal{E}_t, Z_t] &\geq \Pr(X \geq 1 | \mathcal{F}_t, \mathcal{E}_t, Z_t) E \left[\frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_\ell e^{\kappa X}} \mid \mathcal{F}_t, \mathcal{E}_t, Z_t, X \geq 1 \right] \\ &\geq q_\ell (1/q_\ell) (e^{-\kappa \cdot 0} - e^{-\kappa \cdot 1}) = 1 - e^{-\kappa} \end{aligned}$$

So the conditional forward drift is $E [\Delta | \mathcal{F}_t, \mathcal{E}_t] \geq \min\{c\varepsilon, 1 - e^{-\kappa}\}$. Furthermore, κ can be picked in the non-empty interval $(-\ln(1 - c\varepsilon), \delta) \subset (0, \delta)$, so that $1 - e^{-\kappa} > c\varepsilon$ and $E [\Delta | \mathcal{F}_t, \mathcal{E}_t] \geq c\varepsilon$. Next, we compute the *conditional backward drift*, which can be done for the worst case.

$$E [\Delta | \mathcal{F}_t, \bar{\mathcal{E}}_t] \geq -(m - 1) \ln(1 + c\lambda) - \ln(1 + c\lambda) - \sum_{j=1}^m 1/q_j \geq -m(c\lambda + 2/z_*)$$

The probability that event \mathcal{E}_t does not occur is computed as follows. Recall that $X_t^\ell \geq \lceil \gamma_0 \lambda \rceil$ and X_{t+1}^ℓ is binomially distributed random variable with probability at least $z_0(\gamma_0) \geq (1 + \delta)\gamma_0$ by condition (G2), so $E [X_{t+1}^\ell | \mathcal{F}_t] \geq (1 + \delta)\gamma_0 \lambda$. The event $\bar{\mathcal{E}}_t$ happens when the number of individuals at level ℓ is strictly less than $\lceil \gamma_0 \lambda \rceil$ in the next generation. By a Chernoff bound (see [4]), we have

$$\begin{aligned} \Pr(\bar{\mathcal{E}}_t | \mathcal{F}_t) &= \Pr(X_{t+1}^\ell < \lceil \gamma_0 \lambda \rceil | \mathcal{F}_t) \leq \Pr(X_{t+1}^\ell \leq \gamma_0 \lambda | \mathcal{F}_t) \\ &= \Pr(X_{t+1}^\ell \leq (1 - \delta/(1 + \delta))(1 + \delta)\gamma_0 \lambda | \mathcal{F}_t) \\ &\leq \Pr(X_{t+1}^\ell \leq (1 - \delta/(1 + \delta)) E[X_{t+1}^\ell | \mathcal{F}_t] | \mathcal{F}_t) \\ &\leq \exp\left(-\frac{\delta^2 E[X_{t+1}^\ell | \mathcal{F}_t]}{2(1 + \delta)^2}\right) \leq \exp\left(-\frac{\delta^2(1 + \delta)\gamma_0 \lambda}{2(1 + \delta)^2}\right) = e^{-a\lambda} \end{aligned}$$

Recall condition (G3) that $\lambda \geq (2/a) \ln((16m)/(ac\varepsilon z_*))$. This implies that $(8m)/(ac\varepsilon z_*) \leq e^{\frac{a\lambda}{2}}/2 \leq e^{a\lambda}/(a\lambda)$, or $e^{-a\lambda} \leq c\varepsilon z_*/(8m\lambda)$. The drift is therefore

$$\begin{aligned} E [\Delta | \mathcal{F}_t] &= (1 - \Pr(\bar{\mathcal{E}}_t | \mathcal{F}_t)) E [\Delta | \mathcal{F}_t, \mathcal{E}_t] + \Pr(\bar{\mathcal{E}}_t | \mathcal{F}_t) E [\Delta | \mathcal{F}_t, \bar{\mathcal{E}}_t] \\ &= E [\Delta | \mathcal{F}_t, \mathcal{E}_t] - \Pr(\bar{\mathcal{E}}_t | \mathcal{F}_t) (E [\Delta | \mathcal{F}_t, \mathcal{E}_t] - E [\Delta | \mathcal{F}_t, \bar{\mathcal{E}}_t]) \\ &\geq c\varepsilon - \frac{c\varepsilon z_*}{8m\lambda} \left(c\varepsilon + m \left(c\lambda + \frac{2}{z_*} \right) \right) \\ &\geq c\varepsilon - \frac{c\varepsilon}{8} \left(\frac{c\varepsilon z_*}{\lambda m} + z_* c + \frac{2}{\lambda} \right) \geq c\varepsilon - \frac{4c\varepsilon}{8} = \frac{c\varepsilon}{2} \end{aligned}$$

By additive drift [7], $E [T] \leq \frac{2}{c\varepsilon} \left(m\lambda(1 + \ln(1 + c\lambda)) + \sum_{j=1}^m \frac{1}{z_j} \right)$. □

In the special case of unary variation operators Theorem 1 becomes analogous to the main results of [2,8]. It is an open problem whether the upper bound in Theorem 1 is tight. The lack of general tools make this problem hard. The family tree technique [19], the population drift theorem [9], and the fitness level technique in [16], provide lower bounds for population-based algorithms, however only for less general settings than Algorithm 1.

4 Runtime Analysis of Genetic Algorithms

This section provides a version of Theorem 1 tailored to the GAs described in Section 2. The *selective pressure* of a selection mechanism p_{sel} is defined as follows. For any $\gamma \in (0, 1)$ and population P of size λ , let $\beta(\gamma, P)$ be the probability of selecting an individual from P that is at least as good as the individual with rank $\lceil \gamma\lambda \rceil$ (see [2] or [8] for a formal definition). We assume that p_{sel} is *monotone* with respect to fitness values [8], ie for all $P \in \mathcal{X}^\lambda$ and pairs $i, j \in [\lambda]$, $p_{\text{sel}}(i \mid P) \geq p_{\text{sel}}(j \mid P)$ if and only if $f(P(i)) \geq f(P(j))$.

Corollary 1. *Given a function $f : \mathcal{X} \rightarrow \mathbb{R}$ and a partition (A_1, \dots, A_{m+1}) of \mathcal{X} , let $T := \min\{t\lambda \mid |P_t \cap A_{m+1}| > 0\}$ be the runtime of the non-elitist Genetic Algorithm, as described in Section 2, on f . If there exist parameters $s_1, \dots, s_m, s_*, p_0, \varepsilon_1 \in (0, 1]$, $\delta > 0$, and a constant $\gamma_0 \in (0, 1)$ such that for all $j \in [m]$, $P \in \mathcal{X}^\lambda$, and $\gamma \in (0, \gamma_0)$*

$$(C1) \quad p_{\text{mut}}(y \in A_j^+ \mid x \in A_{j-1}^+) \geq s_j \geq s_*$$

$$(C2) \quad p_{\text{mut}}(y \in A_j^+ \mid x \in A_j^+) \geq p_0$$

$$(C3) \quad p_{\text{xor}}(x \in A_j^+ \mid u \in A_{j-1}^+, v \in A_j^+) \geq \varepsilon_1$$

$$(C4) \quad \beta(\gamma, P) \geq \gamma \sqrt{\frac{1+\delta}{p_0 \varepsilon_1 \gamma_0}}$$

$$(C5) \quad \lambda \geq \frac{2}{a} \ln \left(\frac{32mp_0}{(\delta\gamma_0)^2 c s_* \psi} \right) \text{ with } a := \frac{\delta^2 \gamma_0}{2(1+\delta)}, \psi := \min\{\frac{\delta}{2}, \frac{1}{2}\} \text{ and } c := \frac{\psi^4}{24}$$

$$\text{then } E[T] \leq \frac{2}{c\psi} \left(m\lambda(1 + \ln(1 + c\lambda)) + \frac{p_0}{(1+\delta)\gamma_0} \sum_{j=1}^m \frac{1}{s_j} \right).$$

Proof. We show that conditions (C1-5) imply conditions (G1-3) in Theorem 1. We first show that condition (G1) is satisfied for $z_j = \gamma_0(1 + \delta)s_j/p_0$. Assume that $|P \cap A_{j-1}^+| \geq \gamma_0\lambda$. Remark that (C3) written for one level below, which is $p_{\text{xor}}(x \in A_{j-1}^+ \mid u \in A_{j-2}^+, v \in A_{j-1}^+) \geq \varepsilon_1$, implies $p_{\text{xor}}(x \in A_{j-1}^+ \mid u \in A_{j-1}^+, v \in A_{j-1}^+) \geq \varepsilon_1$. To sample an individual in A_j^+ , it suffices that the selection operator picks two individuals u and v from A_{j-1}^+ , that the crossover operator produces an individual x in A_{j-1}^+ from u and v , and the mutation operator produces an individual y in A_j^+ from x . By conditions (C4), (C3) as the remark, and (C1), the probability of this event is at least $\beta(\gamma_0)\beta(\gamma_0)\varepsilon_1 s_j \geq \gamma_0(1 + \delta)s_j/p_0 = z_j$.

We then show that condition (G2) is satisfied. Assume that $|P \cap A_{j-1}^+| \geq \gamma_0\lambda$ and $|P \cap A_j^+| \geq \gamma\lambda$. To produce an individual y in A_j^+ , it suffices that the selection operator picks an individual u in A_{j-1}^+ and an individual v in A_j^+ , that the crossover operator produces an individual x in A_j^+ from u and v , and the mutation operator produces an individual y in A_j^+ from x . By conditions (C4), (C3), and (C2), the probability of this event is at least $\beta(\gamma_0)\beta(\gamma)\varepsilon_1 p_0 \geq (1 + \delta)\gamma$.

Finally, to see that condition (G3) is satisfied, it suffices to note that $az_* = (\delta\gamma_0)^2 s_*/(2p_0)$. Hence, the statement now follows from Theorem 1. \square

4.1 Runtime of GAs on Simple pseudo-Boolean Functions

We apply Corollary 1 to bound the expected runtime of the non-elitist GA on the functions $\text{ONEMAX}(x) := \sum_{i=1}^n x_i$ (also written shortly as $|x|_1$ or OM) and $\text{LEADINGONES}(x) := \sum_{i=1}^n \prod_{j=1}^i x_j$ (shortly as LO).

We first show how to parameterise three standard selection mechanisms such that condition (C4) is satisfied. In *k-tournament selection*, k individuals are sampled uniformly at random with replacement from the population, and the fittest of these individuals is returned. In (μ, λ) -*selection*, parents are sampled uniformly at random among the fittest μ individuals in the population. A function $\alpha : \mathbb{R} \rightarrow \mathbb{R}$ is a ranking function [6] if $\alpha(x) \geq 0$ for all $x \in [0, 1]$, and $\int_0^1 \alpha(x)dx = 1$. In ranking selection with ranking function α , the probability of selecting individuals ranked γ or better is $\int_0^\gamma \alpha(x)dx$. We define *exponential ranking* parameterised by $\eta > 0$ as $\alpha(\gamma) := \eta e^{\eta(1-\gamma)} / (e^\eta - 1)$.¹

Lemma 2. *For any constant $\delta > 0$, there exists a constant $\gamma_0 \in (0, 1)$ such that*

1. *k-tournament selection with $k \geq 4(1 + \delta) / (\varepsilon_1 p_0)$ satisfies (C4)*
2. *(μ, λ) -selection with $\lambda / \mu \geq (1 + \delta) / (\varepsilon_1 p_0)$ satisfies (C4)*
3. *exponential ranking selection with $\eta \geq 4(1 + \delta) / (\varepsilon_1 p_0)$, satisfies (C4).*

Lemma 3 shows that two standard crossover operators satisfy (C3) for $\varepsilon_1 = \frac{1}{2}$.

Lemma 3. *If $x \sim p_{\text{xor}}(u, v)$, where p_{xor} is one-point or uniform crossover, then*

1. *If $\text{LO}(u) = \text{LO}(v) = j$, then $\Pr(\text{LO}(x) \geq j) = 1$.*
2. *If $\text{LO}(u) \neq \text{LO}(v)$, then $\Pr(\text{LO}(x) > \min\{\text{LO}(u), \text{LO}(v)\}) \geq 1/2$.*
3. *$\Pr(\text{OM}(x) \geq \lceil (\text{OM}(u) + \text{OM}(v)) / 2 \rceil) \geq 1/2$.*

Theorem 2. *Assume that the GA with one-point or uniform crossover, bitwise mutation with mutation rate χ/n for a constant $\chi > 0$, and either k-tournament selection with $k \geq 8(1 + \delta)e^\chi$, or (μ, λ) selection with $\lambda/\mu \geq 2(1 + \delta)e^\chi$ or the exponential ranking selection with $\eta \geq 8(1 + \delta)e^\chi$, for a constant $\delta > 0$. Then there exists a constant $c > 0$, such that the GA with population size $\lambda \geq c \ln n$, has expected runtime $O(n\lambda \ln \lambda + n^2)$ on LEADINGONES, and expected runtime $O(n\lambda \ln \lambda)$ on ONEMAX.*

Proof. Let f be either OM or LO. We apply Corollary 1 with the canonical partition of the search space into $n + 1$ levels $A_j := \{x \mid f(x) = j - 1\}$, for $j \in [n + 1]$. We use $p_0 := (1 - \chi/n)^n$ the probability of not flipping any bit position by mutation, and for all $j \in [n]$, define

$$s_j := \begin{cases} (\chi/n)(1 - \chi/n)^{n-1} & \text{if } f = \text{LO, and} \\ (n - j + 1)(\chi/n)(1 - \chi/n)^{n-1}p_0 & \text{if } f = \text{OM.} \end{cases}$$

Considering condition (C1), when $x \in A_j$ it suffices to upgrade x to a higher level, the probability of such an event is at least s_j for LO and $s_j/p_0 > s_j$ for OM.

¹ The proofs of Lemmas 2 and 3 are omitted due to space restrictions.

When $x \in A_j^+$, it suffices to not modify x , the probability of such an event is at least $p_0 \geq s_j$ with sufficiently large n for LO and $p_0 > (n - j + 1)(\chi/n)(1 - \chi/n)^{n-1}p_0 = s_j$ for OM. So, condition (C1) is satisfied for both functions with the given s_j . In addition, condition (C2) is trivially satisfied for the given p_0 and condition (C3) is satisfied for the parameter $\varepsilon_1 := 1/2$ by Lemma 3.

We now look at condition (C4), and remark that $p_0 = (1 - \chi/n)^{(n/\chi-1)\chi}(1 - \chi/n)^\chi \geq e^{-\chi}(1 - \chi/n)^\chi$. So $e^\chi \geq (1 - \chi/n)^\chi/p_0$ and with the given condition for k -tournament, we get $k \geq 8(1 + \delta)e^\chi = 4(1 + \delta)e^\chi/(1/2) \geq 4(1 + \delta)(1 - \chi/n)^\chi/(\varepsilon_1 p_0)$. Then for any constant $\delta' \in (0, \delta)$ and sufficiently large n , literally $n \geq \chi/(1 - ((1 + \delta')/(1 + \delta))^{1/\chi})$, it holds that $k \geq 4(1 + \delta')/(\varepsilon_1 p_0)$. So condition (C4) is satisfied with the constant δ' for k -tournament by Lemma 2. The same reasoning can be applied so that (C4) is also satisfied for the other selection mechanisms.

Finally, $s_* := \min_{j \in [n]} s_j = \Omega(1/n)$. So assuming $\lambda \geq c \ln n$ for a sufficiently large constant c , condition (C5) is satisfied as well. Note that the p_0 part in s_j of LO only removes the p_0 from $p_0/(1 + \delta)\gamma_0$ in the runtime of Corollary 1. Therefore, the upper bounds $O(n\lambda \ln \lambda + n^2)$ and $O(n\lambda \ln \lambda)$ on the expected runtime are proven for OM and LO respectively. \square

Note that the upper bounds in Theorem 2 match the upper bounds obtained in [2] for EAs without crossover.

4.2 Runtime of GAs on the Sorting Problem

Given n distinct elements from a totally ordered set, we consider the problem of finding an ordering of them so that some *measure of sortedness* is maximised. Scharnow et al. [14] considered several sortedness measures in the context of analysing the (1+1) EA. One of those is $INV(\pi)$ which is defined to be the number of pairs (i, j) such that $1 \leq i < j \leq n$, $\pi(i) < \pi(j)$ (i.e. pairs in correct order). We show that with the methods introduced in this paper, analysing GAs on $INV(\pi)$ is not much harder than analysing the (1+1) EA.

As mutation operator, we consider the *Exchange*(π) operator, which consecutively applies N pairwise exchanges between uniformly selected pairs of indices, where N is a random number drawn from a Poisson distribution with parameter 1. We consider a crossover operator, denoted by $p_{xor(p_c)}$, which returns one of the parents unchanged with probability $1 - p_c$. For example, $p_{xor(p_c)}$ is built up from any standard crossover operator so that with probability p_c the standard operator is applied and the offspring is returned, otherwise with probability $1 - p_c$ one of the parents is returned in place of the offspring. This construction corresponds to a typical setting (see [6]) where there is some *crossover probability* p_c of applying the crossover before the mutation. As selection mechanism, we consider k -tournament selection, (μ, λ) -selection, and exponential ranking selection.

Theorem 3. *If the GA uses a $p_{xor(p_c)}$ crossover operator with p_c being any constant in $[0, 1)$, the Exchange mutation operator where the number of exchanges N is drawn from a Poisson distribution with parameter 1, k -tournament selection with $k \geq 8e(1 + \delta)/(1 - p_c)$, or (μ, λ) -selection with $\lambda/\mu \geq 2e(1 + \delta)/(1 - p_c)$,*

or exponential ranking selection with $\eta \geq 8e(1 + \delta)/(1 - p_c)$, then there exists a constant $c > 0$ such that if the population size is $\lambda \geq c \ln n$, the expected time to obtain the optimum of INV is $O(n^2 \lambda \log \lambda)$.

Proof. Define $m := \binom{n}{2}$. We apply Corollary 1 with the canonical partition, $A_j := \{\pi \mid \text{INV}(\pi) = j\}$ for $j \in [m]$. The probability that the exchange operator exchanges 0 pairs is $1/e$. Hence, condition (C2) is trivially satisfied for $p_0 := 1/e$.

To show that condition (C1) is satisfied, define first $s_j := (m - j)p_0/(em)$. In the case that $x \in A_j$, then the probability that the exchange operator exchanges exactly one pair is $1/e$, and the probability that this pair is incorrectly ordered in x , is $(m - j)/m$. In the other case that, $x \in A_j^+$, it is sufficient that the exchange operator exchanges 0 pairs, which by condition (C2) occurs with probability at least $p_0 \geq s_j$. Hence, in both cases, $y \in A_j^+$ with probability at least s_j , and condition (C1) is satisfied.

Condition (C3) is trivially satisfied for $\varepsilon_1 := (1 - p_c)/2$, because the crossover operator returns one of the parents unchanged with probability $1 - p_c$, and with probability $1/2$, this parent is v . Condition (C4) is satisfied for some constant $\gamma_0 \in (0, 1)$ by Lemma 2. Finally, since γ_0, δ , and p_0 are constants, there exists a constant $c > 0$ such that condition (C5) is satisfied for any $\lambda \geq c \ln(n)$.

It therefore follows that the expected runtime of the GA on INV is upper bounded by $O(n^2 \lambda \log \lambda)$. \square

5 Conclusion

Most results in runtime analysis of evolutionary algorithms concern relatively simple algorithms, e.g. the (1+1) EA, which do not employ populations and higher-arity variation operators, such as crossover. This paper introduces a new tool, akin to the fitness-level technique, that easily yields upper bounds on the expected runtime of complex, non-elitist search processes. The tool is illustrated on Genetic Algorithms. Given an appropriate balance between selection and variation operators, we have shown that GAs optimise standard benchmark functions, as well as combinatorial optimisation problems, efficiently. Future applications of the theorem might consider the impact of the crossover operator more in detail, e.g. by a more precise analysis of the population diversity.

Acknowledgements. This research received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 618091 (SAGE), from EPSRC grant no EP/F033214/1 (LANCS), and from Russian Foundation for Basic Research grant 12-01-00122. Early ideas were discussed at Dagstuhl Seminar 13271 Theory of Evolutionary Algorithms.

References

1. Chen, T., Lehre, P., Tang, K., Yao, X.: When is an estimation of distribution algorithm better than an evolutionary algorithm? In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2009), pp. 1470–1477 (2009)

2. Dang, D.C., Lehre, P.K.: Refined upper bounds on the expected runtime of non-elitist populations from fitness levels. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2014) (to appear, 2014)
3. Doerr, B., Doerr, C., Ebel, F.: Lessons from the black-box: Fast crossover-based genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013), pp. 781–788 (2013)
4. Dubhashi, D., Panconesi, A.: Concentration of Measure for the Analysis of Randomized Algorithms. Cambridge University Press, NY (2009)
5. Eremeev, A.V.: Modeling and analysis of genetic algorithm with tournament selection. In: Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M., Ronald, E. (eds.) AE 1999. LNCS, vol. 1829, pp. 84–95. Springer, Heidelberg (2000)
6. Goldberg, D.E.: Genetic Algorithms in search, optimization and machine learning. Addison-Wesley, MA (1989)
7. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence 127(1), 57–85 (2001)
8. Lehre, P.K.: Fitness-levels for non-elitist populations. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2011), pp. 2075–2082 (2011)
9. Lehre, P.K.: Negative drift in populations. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 244–253. Springer, Heidelberg (2010)
10. Lehre, P.K., Yao, X.: Crossover can be constructive when computing unique input-output sequences. Soft Computing 15(9), 1675–1687 (2011)
11. Lehre, P.K., Yao, X.: On the impact of mutation-selection balance on the runtime of evolutionary algorithms. IEEE Transactions on Evolutionary Computation 16(2), 225–241 (2012)
12. Oliveto, P.S., Witt, C.: Improved runtime analysis of the simple genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013), pp. 1621–1628 (2013)
13. Qian, C., Yu, Y., Zhou, Z.H.: An analysis on recombination in multi-objective evolutionary optimization. Artificial Intelligence 204, 99–119 (2013)
14. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. Journal of Mathematical Modelling and Algorithms 3(4), 349–366 (2004)
15. Sudholt, D.: Crossover speeds up building-block assembly. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2012), pp. 689–702 (2012)
16. Sudholt, D.: A new method for lower bounds on the running time of evolutionary algorithms. IEEE Transactions on Evolutionary Computation 17(3), 418–435 (2013)
17. Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge (1999)
18. Wegener, I.: Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. Evolutionary Optimization 48, 349–369 (2002)
19. Witt, C.: Runtime analysis of the $(\mu+1)$ EA on simple pseudo-boolean functions. Evolutionary Computation 14(1), 65–86 (2006)

Maximizing Submodular Functions under Matroid Constraints by Multi-objective Evolutionary Algorithms

Tobias Friedrich¹ and Frank Neumann²

¹ Friedrich-Schiller-Universität Jena, 07743 Jena, Germany

² Optimisation and Logistics, School of Computer Science,
The University of Adelaide, Adelaide, SA 5005, Australia

Abstract. Many combinatorial optimization problems have underlying goal functions that are submodular. The classical goal is to find a good solution for a given submodular function f under a given set of constraints. In this paper, we investigate the runtime of a multi-objective evolutionary algorithm called GSEMO until it has obtained a good approximation for submodular functions. For the case of monotone submodular functions and uniform cardinality constraints we show that GSEMO achieves a $(1 - 1/e)$ -approximation in expected time $\mathcal{O}(n^2 (\log n + k))$, where k is the value of the given constraint. For the case of non-monotone submodular functions with k matroid intersection constraints, we show that GSEMO achieves a $1/(k + 2 + 1/k + \varepsilon)$ -approximation in expected time $\mathcal{O}(n^{k+5} \log(n)/\varepsilon)$.

1 Introduction

Evolutionary algorithms can efficiently find the minima of convex functions. While this is known and well studied in the continuous domain, it is not obvious how an equivalent statement for discrete optimization looks like. Let us recall that a differentiable fitness function $f: \mathbb{R} \rightarrow \mathbb{R}$ is called *convex* if its derivative $\frac{d}{dx}f(x)$ is non-decreasing in x . The bitstring analogue of this is a fitness function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ whose discrete derivative $\partial_i f(x) = f(x + e_i) - f(x)$ is non-decreasing in x for all $1 \leq i \leq n$ with e_i being the i -th unit vector. A discrete function satisfying the aforementioned condition is called *submodular*. Submodularity is the counterpart of convexity in discrete settings [25].

For understanding the properties of continuous optimizers it is central to study their performance for minimizing convex functions. This has been done in detail for continuous evolutionary algorithms [2, 17]. On the other hand, it is rather surprising that there appears to be not a single published study regarding the performance of discrete evolutionary algorithms for optimizing submodular functions. We want to fill this gap and present several approximation results for simple evolutionary algorithms and submodular functions.

Analogous to the situation for convex functions, there is a significant difference between minimization and maximization of submodular functions. Submodular functions can be *minimized* with a (non-trivial) combinatorial algorithm in

polynomial time [19]. On the other hand, submodular function *maximization* is NP-hard as it generalizes many NP-hard combinatorial optimization problems, like maximum cut [10, 15], maximum directed cut [16], maximum facility location [1, 7], and several restricted satisfiability problems [10, 18]. As evolutionary algorithms are especially useful for hard problems, we focus on the maximization of submodular functions.

More formally, we consider the optimization problem $\max\{f(S) : S \in \mathcal{I}\}$, where X is an arbitrary ground set, $f : 2^X \rightarrow \mathbb{R}$ is a fitness function, and $\mathcal{I} \subseteq 2^X$ a collection of independent sets describing the feasible region of the problem. As usual, we assume *value oracle access* to the fitness function; i.e., for a given set S , an algorithm can query an oracle to find its value $f(S)$. We also always assume that the fitness function is *normalized*, i.e., $f(\emptyset) = 0$, and *non-negative*, i.e., $f(A) \geq 0$ for all $A \subseteq X$. We will study the following variants of f and \mathcal{I} :

- *Submodular functions*: A function f is submodular iff $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for all $A, B \subseteq X$.
- *Monotone functions*: A function is monotone iff $f(A) \leq f(B)$ for all $A \subseteq B$.
- *Matroid*: A matroid is a pair (X, \mathcal{I}) composed of a ground set X and a non-empty collection \mathcal{I} of subsets of X satisfying (1) If $A \in \mathcal{I}$ and $B \subseteq A$ then $B \in \mathcal{I}$ and (2) If $A, B \in \mathcal{I}$ and $|A| > |B|$ then $B + x \in \mathcal{I}$ for some $x \in A \setminus B$. The sets in \mathcal{I} are called *independent*, the *rank* of a matroid is the size of any maximal independent set.
- *Uniform matroid*: A uniform matroid (X, \mathcal{I}) of rank $k \in \mathbb{N}$ contains all subsets of size at most k , i.e., $\mathcal{I} = \{A \subseteq X : |A| \leq k\}$.
- *Partition matroid*: A partition matroid is a matroid formed from a direct sum of uniform matroids, i.e., if the universe X is partitioned into k parts X_1, \dots, X_k , then in a partition matroid a set is independent if it contains at most one element from each part.
- *Intersection of k matroids*: Given k matroids $M_1 = (X, \mathcal{I}_1)$, $M_2 = (X, \mathcal{I}_2)$, \dots , $M_k = (X, \mathcal{I}_k)$ on the same ground set X , the intersection of these matroids is the matroid (X, \mathcal{I}) with $\mathcal{I} = \{A \subseteq X \mid A \in \mathcal{I}_i, 1 \leq i \leq k\}$. A simple example for $k = 2$ is the family of matchings in a bipartite graph; or in general the family of hypergraph matchings in a k -partite hypergraph.

Maximizing submodular functions is not only NP-hard, but also NP-hard to approximate. We therefore also have to formalize the notion of an approximation algorithm. We say an algorithm achieves an α -approximation if for all instances of the considered maximization problem, the output returned by the algorithm is at least α times the optimal value.

Our results. Optimizing single objective optimization problems by multi-objective approaches such as the global simple evolutionary multiobjective optimizer (GSEMO) has already been shown to be beneficial for many combinatorial optimization problems [11, 21, 28]. We study GSEMO and prove the following statements.

- Based on the seminal work of Nemhauser, Wolsey, and Fisher [26], we show that GSEMO achieves in polynomial time a $1 - 1/e$ -approximation for maximizing *monotone submodular* functions under a *uniform matroid constraint*.

This approximation factor is optimal in the general setting [27], and it is optimal even for the special case of Max- k -cover, unless $\mathbf{P} = \mathbf{NP}$ [9].

- Based on the more recent work of Lee, Mirrokni, Nagarajan, and Sviridenko [23], we show that GSEMO achieves in polynomial time a $1/(k+2+1/k+\varepsilon)$ -approximation for maximizing *submodular* functions over k *matroid constraints*. Note that this result even holds for *non-monotone* functions.

Outline. The paper is organized as follows. In Section 2, we describe the setting for submodular functions and introduce the algorithm that is subject to our investigations. We analyze the algorithm on monotone submodular functions with a uniform constraint in Section 3 and consider the case of non-monotone submodular functions under matroid constraints in Section 4. Finally, we finish with a discussion on open problems in Section 5.

2 Preliminaries

Optimization of submodular functions and matroids have received a lot of attention in the classical (non-evolutionary) optimization community. For a detailed exposition, we refer to the textbooks of Schrijver [30] and Korte and Vygen [20].

Submodular Functions. When optimizing a submodular function $f: 2^X \rightarrow \mathbb{R}$, we will often consider the incremental value of adding a single element. For this, we denote by $F_A(i) = f(A + i) - f(A)$ the marginal value of i with respect to A . Nemhauser et al. [26, Proposition 2.1] give seven equivalent definitions for submodular functions. Additionally to the definition stated in the introduction we will also use that a function f is submodular iff $F_i(A) \geq F_i(B)$ for all $A \subseteq B \subseteq X$ and $i \in X \setminus B$.

Many common pseudo-Boolean and combinatorial fitness functions are submodular. As we are not aware of any general results for the optimization of submodular function by evolutionary algorithms, we list a few examples of well-known submodular functions:

- *Linear functions:* All linear functions $f: 2^X \rightarrow \mathbb{R}$ with $f(A) = \sum_{i \in A} w_i$ for some weights $w: X \rightarrow \mathbb{R}$ are submodular. If $w_i \geq 0$ for all $i \in X$, then f is also monotone.
- *Cut:* Given a graph $G = (V, E)$ with nonnegative edge weights $w: E \rightarrow \mathbb{R}_{\geq 0}$. Let $\delta(S)$ be the set of all edges that contain both a vertex in S and $V \setminus S$. The cut function $w(\delta(S))$ is submodular but not monotone.
- *Coverage:* Let the ground set be $X = \{1, 2, \dots, n\}$. Given a universe U with n subsets $A_i \subseteq U$ for $i \in X$, and a non-negative weight function $w: U \rightarrow \mathbb{R}_{\geq 0}$. The coverage function $f: 2^X \rightarrow \mathbb{R}$ with $f(S) = |\bigcup_{i \in S} A_i|$ and the weighted coverage function f' with $f'(S) = w(\bigcup_{i \in S} A_i) = \sum_{u \in \bigcup_{i \in S} A_i} w(u)$ are monotone submodular.
- *Rank of a matroid:* The rank function $r(A) = \max\{|S|: S \subseteq A, S \in \mathcal{I}\}$ of a matroid (X, \mathcal{I}) is monotone submodular.

- *Hypervolume Indicator:* Given a set of points in \mathbb{R}^d in the objective space of a multi-objective optimization problem, measure the volume of the space dominated by these points relative to some fixed reference point. The hypervolume is a well-known quality measure in evolutionary multi-objective optimization and is known to be monotone submodular [31].

Matroids. We defined the most important matroids already in the introduction. Matroid theory provides a framework in which many problems from combinatorial optimization can be studied from a unified perspective. Matroids are a special class of so-called *independence systems* that are given by a finite set X and a family of subsets $\mathcal{I} \subseteq X$ such that \mathcal{I} is closed under subsets. Being a matroid is considered to be the property of an independence system which makes greedy algorithms work well. Within evolutionary computation, matroid constraints have been studied only for linear functions [29].

Fitness function. We assume a finite ground set $X = \{x_1, x_2, \dots, x_n\}$ and identify each subset $S \subseteq X$ with a bitstring $x \in \{0, 1\}^n$ such that the i -th bit of x is 1 iff $x_i \in S$. Let $f: \{0, 1\}^n \rightarrow \mathbb{R}_{\geq 0}$ be the given (normalized and non-negative) submodular function and $F \subseteq \{0, 1\}^n$ be the set of feasible solutions. Note, that f is defined on every element of $\{0, 1\}^n$. We set $z(x) = f(x)$ iff $x \in F$ and $z(x) = -1$ iff $x \notin F$ and consider the multi-objective problem

$$g(x) := (z(x), |x|_0),$$

where $|x|_0 = \sum_{i=1}^n (1 - x_i)$ denotes the number of 0-bits in the given bitstring x . We write $g(x) \geq g(y)$ iff $((z(x) \geq z(y)) \wedge (|x|_0 \geq |y|_0))$ holds. If $g(x) \geq g(y)$ holds, we say that y is dominated by x . The solution y is strictly dominated by solution x iff $g(x) \geq g(y)$ and $g(x) \neq g(y)$.

Algorithms. The theoretical runtime analysis of evolutionary algorithms often considers randomized local search (RLS) and the $(1 + 1)$ evolutionary algorithm (EA). The multi-objective counterpart of RLS and $(1+1)$ EA are the simple evolutionary multi-objective optimizer (SEMO) [22] and global SEMO (GSEMO) [12]. Both algorithms have been studied in detail, see [6, 8, 11–13]. We consider the GSEMO given in Algorithm 1.

In the end, we focus on the solution $x^* = \arg \max_{x \in P} z(x)$ and study the quality of this solution. We study the expected number of iterations (of the repeat loop) of GSEMO until x^* is an α -approximation of an optimal solution OPT, i.e. $f(x^*)/\text{OPT} \geq \alpha$ holds. Here α denotes the investigated approximation ratio for the considered problem. We call the expected number of iterations to reach an α -approximation, the expected (run)time to achieve an α -approximation.

3 Monotone Submodular Functions with a Uniform Constraint

In this section, we investigate submodular functions with one uniform constraint. In the case of one uniform constraint of size k , a solution $x \in X$ is feasible if it has at most k elements. Hence, we have $F = \{x \mid x \in X \wedge |x|_1 \leq k\}$.

Algorithm 1. GSEMO Algorithm

```

1 choose  $x \in \{0, 1\}^n$  uniformly at random
2 determine  $g(x)$ 
3  $P \leftarrow \{x\}$ 
4 repeat
5     choose  $x \in P$  uniformly at random
6     create  $x'$  by flipping each bit  $x_i$  of  $x$  with probability  $1/n$ 
7     determine  $g(x')$ 
8     if  $x'$  is not strictly dominated by any other search point in  $P$  then
9         include  $x'$  into  $P$ 
10        delete all other solutions  $z \in P$  with  $g(z) \leq g(x')$  from  $P$ 
11 until stop

```

Theorem 1. *The expected time until GSEMO has obtained a $(1 - \frac{1}{e})$ -approximation for a monotone submodular function f under a uniform constraint of size k is $\mathcal{O}(n^2 (\log n + k))$.*

Proof. We first study the expected time until GSEMO has produced the solution 0^n for the first time. This solution is Pareto optimal and will therefore stay in the population after it has been produced for the first time. Furthermore, the population size is upper bounded by $n + 1$ as it contains for each i , $0 \leq i \leq n$ at most one solution having exactly i 1-bits. The solution 0^n is feasible and has the maximum number of 0-bits. This implies that the population will not include any infeasible solution to the submodular function f after having included 0^n .

For this step, we consider in each iteration the individual y that has the minimum number of 1-bit among all individuals in the population and denote $\ell = |y|_1$ the number of 1-bits in this individual. Note, that ℓ can not increase during the run of the algorithm. For $1 < \ell \leq n$ a solution y' with $|y'|_1 = \ell - 1$ is produced with probability at least $\ell/(en^2)$ as y' can be produced by selecting y for mutation and flipping one of the ℓ 1-bits. The expected waiting time to include the solution 0^n for the first time into the population is therefore upper bounded by $\sum_{\ell=1}^n (\frac{\ell}{en^2})^{-1} = \mathcal{O}(n^2 \log n)$.

For the remainder of the proof, we follow the ideas of the proof for the greedy algorithm in Nemhauser et al. [26]. We show that GSEMO produces in expected time $\mathcal{O}(n^2 k)$ for each $0 \leq j \leq k$ a solution X_j with

$$f(X_j) \geq \left(1 - \left(1 - \frac{1}{k}\right)^j\right) \cdot f(\text{OPT}), \tag{1}$$

where $f(\text{OPT})$ denotes the value of a feasible optimal solution. Note, that a solution is feasible iff it has at most k 1-bits. After having including the solution 0^n into the population this is true for $j = 0$. The proof is done by induction. Assume that GSEMO has already obtained a solution fulfilling Equation 1 for each j , $0 \leq j \leq i < k$. We claim that choosing the solution $x \in P$ with $|x|_1 = i$ for mutation and inserting the element corresponding to the largest possible

increase of f increases the value of f by at least $\delta_{i+1} \geq \frac{1}{k} \cdot (f(\text{OPT}) - f(X_i))$. Let δ_{i+1} be the increase in f that we obtain when choosing the solution $x \in P$ with $|x|_1 = i$ for mutation and inserting the element corresponding to the largest possible increase.

Due to monotonicity and submodularity, we have $f(\text{OPT}) \leq f(X_i \cup \text{OPT}) \leq f(X_i) + k\delta_{i+1}$ which implies $\delta_{i+1} \geq \frac{1}{k} \cdot (f(\text{OPT}) - f(X_i))$. This leads to $f(X_{i+1}) \geq f(X_i) + \frac{1}{k} (f(\text{OPT}) - f(X_i)) \geq \left(1 - \left(1 - \frac{1}{k}\right)^{i+1}\right) \cdot f(\text{OPT})$.

For $i = k$, we get $\left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot f(\text{OPT}) \geq \left(1 - \frac{1}{e}\right) f(\text{OPT})$. The probability for such a step going from i to $i + 1$ is lower bounded by $\frac{1}{en^2}$ and hence the expected time until a $\left(1 - \frac{1}{e}\right)$ -approximation has been obtained is at most

$$\mathcal{O}(n^2 \log n) + \sum_{i=0}^k \left(\frac{1}{en^2}\right)^{-1} = \mathcal{O}(n^2 (\log n + k)). \quad \square$$

Max- k -Cover. Let us demonstrate the applicability of Theorem 1 by two examples. First, reconsider the maximum coverage problem introduced in Section 2. Given a universe U with subsets $A_1, A_2, \dots, A_n \subseteq U$, we want to maximize a coverage function $f(S) = |\bigcup_{i \in S} A_i|$ such that $|S| \leq k$. Theorem 1 immediately implies:

Corollary 1. *The expected time until the GSEMO has obtained a $(1 - 1/e)$ -approximation for the Max- k -Cover problem is $\mathcal{O}(n^2 (\log n + k))$. The achieved approximation factor is optimal, unless $\text{P} = \text{NP}$ [9].*

Hypervolume indicator. As a second example, we consider a problem from evolutionary multiobjective optimization. As discussed in Section 2, the hypervolume indicator is a monotone submodular function. The hypervolume subset selection problem (HYP-SSP), where we are given n points in \mathbb{R}^d and want to select a subset of size k with maximal hypervolume [4, 5, 14], therefore aims at maximizing a monotone submodular function $f: \{0, 1\}^n \rightarrow \mathbb{R}_{\geq 0}$ under a uniform matroid constraint of rank k . Theorem 1 implies therefore:

Corollary 2. *The expected time until the GSEMO has obtained a $(1 - 1/e)$ -approximation for HYP-SSP is $\mathcal{O}(n^2 (\log n + k))$.*

For dimensions $d > 2$ this is significantly faster than the best known exact algorithm with runtime $\mathcal{O}(n^k)$ [3]. Note that HYP-SSP can be solved in time $\mathcal{O}(n(k + \log n))$ for $d = 2$ [4, 5].

4 Non-monotone Submodular Functions under Matroid Constraints

We now turn to submodular functions that are not necessarily monotone. The constraints are given by k matroids. Given k arbitrary matroids M_1, \dots, M_k defined on a ground set X together with their independent systems I_1, \dots, I_k .

We consider the problem $\max \left\{ f(x) \mid x \in F := \bigcap_{j=1}^k I_j \right\}$, where f is a non-negative submodular function defined on the ground set X . Note that this setting is much more general than the one investigated in the previous section.

For our analysis, we make use of the following lemma in [23].

Lemma 1. *Let x be a solution such that no solution with fitness at least $(1 + \frac{\epsilon}{n^4}) \cdot f(x)$ can be achieved by deleting one element or by inserting k elements and deleting one element. Then x is a $(\frac{1}{k+2+\frac{1}{k}+\epsilon})$ -approximation.*

Lemma 1 states that there is always the possibility to achieve a certain progress if no good approximation has been obtained. We use this to show the following results for GSEMO.

Theorem 2. *The expected time until the GSEMO has obtained a $(\frac{1}{k+2+\frac{1}{k}+\epsilon})$ -approximation for any (non necessarily) non-monotone submodular function under k matroid constraints is $\mathcal{O}(\frac{1}{\epsilon} n^{k+5} \log n)$.*

Proof. Following previous investigations, GSEMO introduces the solution 0^n in the population after an expected number of $\mathcal{O}(n^2 \log n)$ steps. This solution is Pareto optimal and will from that point on stay in the population. Furthermore, 0^n is a feasible solution and has the largest possible number of 0-bits. Hence, from the time 0^n has been included in the population, the population will never include infeasible solutions.

Selecting 0^n for mutation and inserting the element that leads to the largest increase in the f -value produces a solution y with $f(y) \geq \text{OPT}/n$. The reason for this is that the number of elements is limited by n and that f is submodular. Having obtained a solution of fitness at least OPT/n , we focus in each iteration on the individual having the largest f -value in P . Due to the selection mechanism of GSEMO a solution with the maximal f -value will always stay in the population and the value will not decrease during the run of the algorithm.

As long as the algorithm has not obtained a solution of the desired quality, it can produce from its current solution x a feasible offspring y such that $f(y) \geq (1 + \frac{\epsilon}{n^4}) \cdot f(x)$. The expected waiting time for this event is $\mathcal{O}(n^{k+1})$ as at most $k + 1$ specific bits have to be flipped.

Starting with a solution of quality at least OPT/n the number of such steps in order to achieve an optimal solution is upper bounded by $\log_{1+\frac{\epsilon}{n^4}} \frac{\text{OPT}}{\text{OPT}/n} = \mathcal{O}(\frac{1}{\epsilon} n^4 \log n)$. Hence, the expected time to achieve a $(\frac{1}{k+2+\frac{1}{k}+\epsilon})$ -approximation is $\mathcal{O}(\frac{1}{\epsilon} n^{k+5} \log n)$. □

As an example, let us consider again the NP-complete Maximum Cut problem, where for a given graph $G = (V, E)$ with n vertices and nonnegative edge weights $w: E \rightarrow \mathbb{R}_{\geq 0}$, we want to maximize the cut function $\delta(S)$ over all $S \subseteq V$ as defined in Section 2. It is known that the greedy algorithm achieves a 0.5-approximation while the best known algorithms achieve a 0.87856-approximation [15]. Theorem 2 immediately implies the following.

Corollary 3. *The expected time until the GSEMO has obtained a $1/(4 + \varepsilon)$ -approximation for the Maximum Cut problem is $\mathcal{O}(\frac{1}{\varepsilon}n^6 \log n)$.*

Note that this result is presumably not tight. We conjecture that a less general analysis can show that GSEMO achieves a $1/2$ -approximation.

5 Discussion and Open Problems

Maximizing submodular functions under matroid constraints is a very general optimization problem which contains many classical combinatorial optimization problems like maximum cut [10, 15], maximum directed cut [16], maximum facility location [1, 7], and others. We presented a number of positive results for the approximation behavior of the GSEMO algorithm in the framework. To the best of our knowledge, this is the first paper on the analysis of evolutionary algorithms optimizing *submodular functions*. The only result on the performance of evolutionary algorithms under *matroid constraints* is by Reichel and Skutella [29]. They showed that a $(1+1)$ -EA achieves in polynomial time a $1/k$ -approximation for maximizing a linear function subject to k matroid constraints.

This paper gives a first set of results, but also leaves many questions open. We briefly name a few:

- We only study the SEMO algorithm, but similar results might be possible for population-based algorithms with appropriate diversity measures.
- Our runtime upper bounds might not be tight. It would be interesting to show matching lower bounds, especially for comparing different algorithms and function classes.
- The proven approximation guarantees for SEMO hold for very general problem classes. Much tighter results should be possible for specific problems like Maximum Cut.
- For RLS and $(1+1)$ -EA we conjecture an exponential runtime lower bound to obtain the same approximation ratio for maximizing (monotone) submodular function if the $(1+1)$ -EA starts at a random (feasible) solution.
- Minimizing submodular functions is in general simpler than maximizing submodular functions. However, it is not obvious what this implies for evolutionary algorithms minimizing submodular functions.
- Our proofs strongly rely on the greedy-like behavior of SEMO. It might either be possible (i) to prove a general relationship between SEMO and greedy algorithms or (ii) to give an example where SEMO strictly outperforms a greedy strategy.
- We assume value oracle access to the fitness function f . It might be worth studying the black box complexity of submodular functions in the sense of Lehre and Witt [24].

Acknowledgments. The research leading to these results has received funding from the Australian Research Council (ARC) under grant agreement DP140103400 and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 618091 (SAGE).

References

- [1] Ageev, A.A., Sviridenko, M.: An 0.828-approximation algorithm for the uncapacitated facility location problem. *Discrete Applied Mathematics* 93(2-3), 149–156 (1999)
- [2] Beyer, H.-G., Schwefel, H.-P.: Evolution strategies – a comprehensive introduction. *Natural Computing* 1(1), 3–52 (2002)
- [3] Bringmann, K., Friedrich, T.: An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation* 18(3), 383–402 (2010)
- [4] Bringmann, K., Friedrich, T., Klitzke, P.: Generic postprocessing via subset selection for hypervolume and epsilon-indicator. In: Bartz-Beielstein, T., et al. (eds.) PPSN XIII 2014. LNCS, vol. 8672, pp. 518–527. Springer, Heidelberg (2014)
- [5] Bringmann, K., Friedrich, T., Klitzke, P.: Two-dimensional subset selection for hypervolume and epsilon-indicator. In: Annual Conference on Genetic and Evolutionary Computation (GECCO). ACM Press (2014b)
- [6] Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: On the effects of adding objectives to plateau functions. *IEEE Transactions on Evolutionary Computation* 13(3), 591–603 (2009)
- [7] Cornuejols, G., Fisher, M., Nemhauser, G.L.: On the uncapacitated location problem. In: *Studies in Integer Programming*. Annals of Discrete Mathematics, vol. 1, pp. 163–177. Elsevier (1977)
- [8] Doerr, B., Kodric, B., Voigt, M.: Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 432–439 (2013)
- [9] Feige, U.: A threshold of $\ln n$ for approximating set cover. *J. ACM* 45(4), 634–652 (1998)
- [10] Feige, U., Goemans, M.X.: Approximating the value of two power proof systems, with applications to MAX 2SAT and MAX DICUT. In: *3rd Israel Symposium on Theory and Computing Systems (ISTCS)*, pp. 182–189 (1995)
- [11] Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation* 18(4), 617–633 (2010)
- [12] Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1918–1925 (2003)
- [13] Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. *Evolutionary Computation* 18(3), 335–356 (2010)
- [14] Glasmachers, T.: Optimized approximation sets of low-dimensional benchmark pareto fronts. In: Bartz-Beielstein, T., et al. (eds.) PPSN XIII 2014. LNCS, vol. 8672, pp. 569–578. Springer, Heidelberg (2014)
- [15] Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* 42(6), 1115–1145 (1995)
- [16] Halperin, E., Zwick, U.: Combinatorial approximation algorithms for the maximum directed cut problem. In: *Twelfth Annual Symposium on Discrete Algorithms (SODA)*, pp. 1–7 (2001)
- [17] Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., Larranaga, P., Inza, I., Bengoetxea, E. (eds.) *Towards a New Evolutionary Computation*. Advances in estimation of distribution algorithms, pp. 75–102. Springer (2006)
- [18] Håstad, J.: Some optimal inapproximability results. *J. ACM* 48(4), 798–859 (2001)

- [19] Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM* 48(4), 761–777 (2001)
- [20] Korte, B., Vygen, J.: *Combinatorial Optimization: Theory and Algorithms*, 4th edn. Springer (2007)
- [21] Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. *Algorithmica* 65(4), 754–771 (2013)
- [22] Laumanns, M., Thiele, L., Zitzler, E., Welzl, E., Deb, K.: Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) *PPSN 2002. LNCS*, vol. 2439, pp. 44–53. Springer, Heidelberg (2002)
- [23] Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Non-monotone submodular maximization under matroid and knapsack constraints. In: *Forty-First Annual ACM Symposium on Theory of Computing (STOC)*, pp. 323–332 (2009)
- [24] Lehre, P.K., Witt, C.: Black-box search by unbiased variation. *Algorithmica* 64(4), 623–642 (2012)
- [25] Lovász, L.: Submodular functions and convexity. In: Bachem, A., Korte, B., Grötschel, M. (eds.) *Mathematical Programming: The State of the Art*. Springer (1983)
- [26] Nemhauser, G., Wolsey, L., Fisher, M.: An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming* 14(1), 265–294 (1978)
- [27] Nemhauser, G.L., Wolsey, L.A.: Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research* 3(3), 177–188 (1978)
- [28] Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. *Natural Computing* 5(3), 305–319 (2006)
- [29] Reichel, J., Skutella, M.: Evolutionary algorithms and matroid optimization problems. *Algorithmica* 57(1), 187–206 (2010)
- [30] Schrijver, A.: *Combinatorial Optimization – Polyhedra and Efficiency*. Springer (2003)
- [31] Ulrich, T., Thiele, L.: Bounding the effectiveness of hypervolume-based $(\mu + \lambda)$ -archiving algorithms. In: Hamadi, Y., Schoenauer, M. (eds.) *LION 2012. LNCS*, vol. 7219, pp. 235–249. Springer, Heidelberg (2012)

On the Runtime Analysis of Fitness Sharing Mechanisms^{*}

Pietro S. Oliveto¹, Dirk Sudholt¹, and Christine Zarges²

¹ University of Sheffield, Sheffield, UK

² University of Birmingham, Birmingham, UK

Abstract. Fitness sharing is a popular diversity mechanism implementing the idea that similar individuals in the population have to share resources and thus, share their fitnesses. Previous runtime analyses of fitness sharing studied a variant where selection was based on populations instead of individuals. We use runtime analysis to highlight the benefits and dangers of the original fitness sharing mechanism on the well-known test problem TWOMAX, where diversity is crucial for finding both optima. In contrast to population-based sharing, a $(2+1)$ EA in the original setting does not guarantee finding both optima in polynomial time; however, a $(\mu+1)$ EA with $\mu \geq 3$ always succeeds in expected polynomial time. We further show theoretically and empirically that large offspring populations in $(\mu+\lambda)$ EAs can be detrimental as overpopulation can make clusters of search points go extinct.

Keywords: Evolutionary computation, diversity mechanisms, fitness sharing, runtime analysis.

1 Introduction

Diversity mechanisms are used in evolutionary computation to tackle multimodal optimisation problems [7]. The main idea is to maintain dissimilar individuals in the population such that different *niches* explore different peaks of the fitness landscape. A popular diversity mechanism is *fitness sharing* [1,6]. In this scheme niche formation is induced by using a *sharing function* that derates the fitness of an individual by an amount related to its similarity to the rest of the population. Different fitness sharing functions are obtained according to how the distance between individuals is defined [7]. *Genotypic sharing* uses Hamming distance and is generally employed when no phenotypic knowledge is available [7]. In *phenotypic sharing* the distance is defined using problem specific knowledge.

Previous theoretical work on diversity mechanisms has concentrated on a somewhat unusual implementation of the sharing mechanism. Let P denote the union of parents and offspring. Rather than selecting *individuals* based on their

* The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 618091 (SAGE).

shared fitness $f(x, P)$, selection was done on a level of populations, creating a population that maximises the overall shared fitness of the population (i. e., creating $P^* = \arg \max\{\sum_{x \in P'} f(x, P') \mid P' \subset P, |P'| = \mu\}$ [3,4,10,11]). The drawback of this approach is that all the possible subsets of P of size μ need to be examined. For large μ and λ , this is prohibitive.

In this paper we analyse the performance of the original fitness sharing approach used in many practical applications [5]. We consider a standard $(\mu+\lambda)$ EA (see Algorithm 1) using the shared fitness values within the selection for replacement. We analyse the algorithm on the same TWOMAX function used in the literature for the analysis of the effectiveness of the previous approach for multimodal optimisation [4].

$\text{TWOMAX}(x) := \max\{\sum_{i=1}^n x_i, n - \sum_{i=1}^n x_i\}$ is a simple bimodal function consisting of two different symmetric branches (i. e., ONEMAX and ZEROMAX) and we have defined both 0^n and 1^n to be global optima. Since we aim at analysing the global exploration capabilities of the population-based EA, we call a run *successful* if it manages to find both optima (i. e., a population is reached that contains both 0^n and 1^n) efficiently. The expected number of generations for this to happen is called *expected running time*. Apart from TWOMAX being the ideal benchmark function for the analysis (i. e., the simplest bimodal function), its choice also allows comparisons with the previous approaches.

A $(\mu+1)$ EA using the unconventional approach (i. e., maximising the phenotypic shared fitness of the population) can efficiently optimise TWOMAX for any population size $\mu \geq 2$ [4]. The reason is that, in any population, the individuals with the smallest and the largest number of ones are always accepted for the next generation. Our analysis shows that using the conventional (phenotypic) sharing approach leads to considerably different behaviours of evolutionary algorithms. We illustrate this by using the analytical framework presented in Sect. 2.

We first concentrate on the effects of the parent population in Sect. 3. A population of size $\mu = 2$ is not sufficient to guarantee that the $(\mu+1)$ EA finds both optima in polynomial time. If the two individuals are initialised on the same branch, then there is a high probability that they will both find the same local optimum. Furthermore, there is a chance that the algorithm fails also when the two individuals are initialised on opposite branches. This leads to a worse failure probability than that of a simple crowding algorithm or that of a $(1+1)$ EA that is restarted twice. On the other hand Sect. 4 shows that for $\mu \geq 3$, once the population is close enough to one optimum, individuals descending the branch heading towards the other optimum are accepted. This threshold, that allows successful runs with probability 1, lies further away from the local optimum as the population size increases.

Concerning the effects of the offspring population, in Sect. 5 we show that large values of λ can be detrimental. We rigorously prove that increasing the offspring population of a $(\mu+1)$ EA to a $(\mu+\lambda)$ EA, with $\mu = 2$ and $\lambda \geq 2$ a constant, results in an overcrowding that can make a (sub-)population go extinct. For the special case of $\lambda = 2$ we also prove an increased failure probability.

Algorithm 1. $(\mu+\lambda)$ EA with fitness sharing

-
1. Choose μ individuals uniformly at random from $\{0, 1\}^n$.
 2. **repeat**
 3. **for** each $1 \leq i \leq \lambda$ **do**
 4. Select a parent x uniformly at random from the population.
 5. Let $x^i := x$. Flip each bit in x^i independently with probability $1/n$.
 6. **end for**
 7. Create a new population by selecting the μ best individuals according to their shared fitness, breaking ties towards favouring offspring over parents, breaking remaining ties uniformly at random.
 8. **until** stopping criterion met
-

We complement this result with an empirical analysis that suggests that the $(\mu+1)$ EA is successful if $\lambda < \lfloor \mu/2 \rfloor$ and that it almost always fails for $\lambda \geq \mu$.

In this extended abstract, some proofs are omitted due to space limitations.

2 Analytical Framework

Throughout this work, $|x|$ denotes the number of 1-bits in x . The shared fitness of an individual x in population P is $f(x, P) := \frac{f(x)}{\sum_{y \in P} sh(x, y)}$ and the sharing function is $sh(x, y) := \max\{0, 1 - (d(x, y)/\sigma)^\alpha\}$. Here, d is the distance between the two individuals, σ is the *sharing distance* beyond which individuals do not share fitness and α is a constant, typically set to 1, that regulates the shape of the sharing function. We consider fitness sharing with phenotypic sharing as in [4], where the distance between individuals is based on the number of ones. We use $\sigma = n/2$ (as in [4]) and the standard value $\alpha = 1$ and obtain

$$f(x, P) := \frac{f(x)}{\sum_{y \in P} \max\left\{0, 1 - \frac{\|x - y\|}{n/2}\right\}}.$$

Let $P := \{x_1, x_2, \dots, x_s\}$ denote the extended population of current search points and the new offspring, labelled such that $|x_1| \leq |x_2| \leq \dots \leq |x_s|$. Let $D_j := \sum_{i=1}^s \min\{\|x_j - x_i\|, n/2\}$ denote the sum of phenotypic distances to all other members of the extended population. Individual distances are capped at the sharing distance $n/2$ so that the shared fitness can be written as

$$f(x_i, P) = \frac{f(x_i)}{s - 2D_i/n}.$$

Phenotypic fitness sharing, along with the shape of the TWOMAX function, implies that a unique best individual will always survive, as it has a better fitness than the individual with the closest number of ones, and it has a larger phenotypic distance to other individuals. This means that in a $(\mu+1)$ EA the current best fitness never decreases; this also holds if multiple individuals have the same current best fitness, as only one individual is removed by selection.

Lemma 1. *Let $P = \{x_1, \dots, x_s\}$ with $|x_1| \leq \dots \leq |x_s|$. If $f(x_1) > f(x_2)$ then $f(x_1, P) > f(x_2, P)$. Likewise, if $f(x_{s-1}) < f(x_s)$ then $f(x_{s-1}, P) < f(x_s, P)$.*

As a result, the $(\mu+1)$ EA never decreases its current best fitness and finds at least one optimum in expected time $O(\mu n \log n)$.

The time bound follows from standard arguments, as used in [4]. The symmetry between $f(x_1, P)$ vs. $f(x_2, P)$ and $f(x_{s-1}, P)$ vs. $f(x_s, P)$ follows from swapping the meaning of zeros and ones. This also applies to further statements, where for simplicity we omit symmetric statements.

The following Main Lemma gives sufficient and necessary conditions on when the shared fitness of one individual is better than another.

Lemma 2 (Main Lemma). *Let $P = \{x_1, \dots, x_s\}$ with $|x_1| \leq \dots \leq |x_s|$ and fix $1 \leq i \leq s - 1$. If $f(x_i) - f(x_{i+1}) = |x_{i+1}| - |x_i| > 0$ and $|x_s| - |x_1| \leq n/2$,*

$$\begin{aligned} f(x_{i+1}, P) \geq f(x_i, P) &\Leftrightarrow f(x_i) \cdot (2i - s) + D_i \geq s \cdot n/2 \\ &\Leftrightarrow f(x_{i+1}) \cdot (2i - s) + D_{i+1} \geq s \cdot n/2. \end{aligned}$$

The same holds if all inequalities “ \geq ” are replaced by strict inequalities “ $>$ ”. Moreover, for $i = s - 1$

$$f(x_s, P) > f(x_{s-1}, P) \Leftrightarrow |x_s| > \sum_{i=1}^{s-1} |x_i| - n/2 \cdot (s - 4).$$

Proof. Note that $|x_s| - |x_1| \leq n/2$ implies that all pairs of individuals do share fitness. Comparing D_i and D_{i+1} , for the latter the distance to x_1, \dots, x_{i-1} is higher by $|x_{i+1}| - |x_i|$, and the distance to x_{i+2}, \dots, x_s is lower by $|x_{i+1}| - |x_i|$:

$$\begin{aligned} D_{i+1} &= D_i + (i - 1) \cdot (|x_{i+1}| - |x_i|) + (s - i - 1) \cdot (|x_i| - |x_{i+1}|) \\ &= D_i + (2i - s) \cdot (|x_{i+1}| - |x_i|). \end{aligned}$$

Using the shorthand $h := |x_{i+1}| - |x_i|$,

$$f(x_{i+1}, P) = \frac{f(x_{i+1})}{s - \frac{D_{i+1}}{n/2}} = \frac{f(x_i) - h}{s - \frac{D_i + (2i-s)h}{n/2}}.$$

Now $f(x_{i+1}, P) \geq f(x_i, P)$ is equivalent to

$$\begin{aligned} \frac{f(x_i) - h}{s - \frac{D_i + (2i-s)h}{n/2}} &\geq \frac{f(x_i)}{s - \frac{D_i}{n/2}} \\ \Leftrightarrow (f(x_i) - h) \cdot (sn/2 - D_i) &\geq f(x_i) \cdot (sn/2 - D_i - (2i - s)h) \\ \Leftrightarrow f(x_i) \cdot (2i - s)h + h \cdot D_i &\geq h \cdot sn/2 \\ \Leftrightarrow f(x_i) \cdot (2i - s) + D_i &\geq sn/2. \end{aligned}$$

In the last step we used $h > 0$. The same calculations hold if “ \geq ” is replaced by “ $>$ ” throughout. The second equivalence from the statement follows from

$$\begin{aligned} f(x_i) \cdot (2i - s) + D_i &= (f(x_{i+1}) + h) \cdot (2i - s) + D_{i+1} - h(2i - s) \\ &= f(x_{i+1}) \cdot (2i - s) + D_{i+1}. \end{aligned}$$

The last statement follows from simple manipulations. □

The Main Lemma gives a condition for the individual of lowest raw fitness (i. e., x_s) to be accepted by selection. Concerning the $(\mu+1)$ EA, the condition clearly shows that for $\mu = 2$ at least $n/2$ bits have to flip (i. e., $|x_3| - |x_2| \geq n/2$). On the other hand, for $\mu \geq 3$ offspring with lower fitness values are accepted once the population is close enough to the optimum 0^n . This threshold is further away from the optimum as the population size increases. If mutation was only allowed to flip one bit and $\mu = 3$, then it is necessary that both x_1 and x_2 reach the local optimum before decreasing moves are accepted (i. e., $|x_1| + |x_2| = 0$). For $\mu = 4$ the sum of 1-bits in the first 4 individuals can be up to $|x_1| + |x_2| + |x_3| + |x_4| \leq n/2$ for any decreasing move to be accepted by the $(\mu+1)$ EA.

In general, the conditions from Lemma 2 are true for x_{s-1} and x_s if $|x_{s-1}| < n/2$ and two individuals are in the optimum 0^n as then

$$f(x_{s-1})(s-2) + D_{s-1} \geq (n - |x_{s-1}|)(s-2) + (s-2)|x_{s-1}| - \sum_{i=1}^{s-2} |x_i| > n(s-2) - (s-4)n/2 = sn/2.$$

Lemma 3. *If $P = \{x_1, \dots, x_s\}$, $|x_1| \leq \dots \leq |x_s|$, with $|x_{s-1}| < n/2$ and $|x_1| = |x_2| = 0$ then $f(x_{s-1}, P)(s-2) + D_{s-1} > sn/2$.*

3 Population Size $\mu = 2$ Is Not Enough

We first investigate the $(2+1)$ EA, showing that a population size of $\mu = 2$ is not sufficient to guarantee finding both optima.

The following lemma gives sufficient and necessary conditions for a single individual on a branch to survive. For $|x_3| = |x_2|$ the statement implies that x_1 survives if the distance from $n/2$ to x_2 is less than around $3/2$ the distance from $n/2$ to x_1 . The condition for survival sharpens when $|x_3| > |x_2|$; however, as x_2 and x_3 result from a mutation of one another, $|x_3| - |x_2|$ is bounded from above by the number of bits flipped in that mutation.

Lemma 4. *Let $\mu = 2$ and $P = \{x_1, x_2, x_3\}$ with $|x_1| < n/2 < |x_2| \leq |x_3|$ and $|x_3| - |x_1| \leq n/2$. Let $d_1 := n/2 - |x_1|$ and $d_2 := |x_2| - n/2$, then*

$$f(x_1, P) > f(x_2, P) \Leftrightarrow d_2 < \left(\frac{3}{2} + \frac{7d_1}{n + 6|x_1|} \right) \cdot d_1 + \frac{(|x_3| + |x_2|)(f(x_2) - f(x_1))}{n/2 + 3|x_1|}.$$

The following theorem states that with a probability greater than $1/2$, the $(2+1)$ EA will end up with both individuals in the same optimum, leading to an exponential running time from there. This performance is worse than having two independent runs of a $(1+1)$ EA, as in deterministic crowding, for which the probability of finding both optima is exactly $1/2$ [4].

Theorem 1. *The $(2+1)$ EA with fitness sharing with probability $1/2 + \Omega(1)$ will reach a population with both members in the same optimum, and then the expected time for finding both optima from there is $\Omega(n^{n/2})$.*

Proof. Using that $2^{-n} \binom{n}{i} \leq 2^{-n} \binom{n}{n/2} = \Theta(1/\sqrt{n})$ for any $0 \leq i \leq n$, it is easy to show that with probability $1 - O(n^{1/3}/\sqrt{n}) = 1 - o(1)$ for both initial search points x_1, x_2 we have $|x_1|, |x_2| \notin [n/2 - n^{1/3}, n/2 + n^{1/3}]$. By symmetry, with probability $1/2 - o(1)$, x_1 and x_2 are on the same branch. The probability of a mutation jumping from one branch to the other is then at most $1/(n^{1/3!}) = 2^{-\Omega(n^{1/3} \log n)}$, and the probability of this happening in expected polynomial time is still of the same order. This implies that w. o. p. no individuals on the opposite branch will be created in polynomial time as long as no offspring of decreasing fitness are ever accepted on the branch. In the following we prove by contradiction that such offspring are always rejected.

Assuming both search points and the offspring are all on the same branch, w. l. o. g. the left branch, by Lemma 2

$$f(x_3, P) \geq f(x_2, P) \Leftrightarrow f(x_2) + D_2 \geq 3 \cdot \frac{n}{2} \tag{1}$$

where $D_2 = (|x_2| - |x_1|) + (|x_3| - |x_2|) = |x_3| - |x_1|$. Then $f(x_2) + D_2 = n - |x_2| + |x_3| - |x_1| \leq n + |x_3| - |x_2|$. This implies that (1) only holds if $|x_3| - |x_2| \geq n/2$, which is a contradiction since there are no points on the left branch differing in more than $n/2$ one-bits. Hence, the claim that no offspring on the left branch of worse fitness than x_2 are ever accepted, is proved. By Lemma 1, 0^n will be reached in expected time $O(n \log n)$. In a further expected $2 \cdot (1 - 1/n)^n = O(1)$ generations, the extended population will contain a clone of 0^n , and from then on any offspring x_3 with $0 < |x_3| \leq n/2$ will be rejected. Then the expected time to create an individual on the other branch is $\Omega(n^{n/2})$ since at least $n/2$ bits need to flip.

The claimed probability $1/2 + \Omega(1)$ follows from considering the following additional event, which is disjoint from the above. The algorithm also fails if, using the notation from Lemma 4, $3\sqrt{n}/4 \leq d_2 \leq \sqrt{n}$ (probability at least 0.02) and $\sqrt{n}/3 \geq d_1 \geq 0$ (probability at least 0.21). If then in the first generation a clone of x_2 is generated (probability at least $1/2 \cdot (1 - 1/n)^n > 1/8$), we have

$$\left(\frac{3}{2} + \frac{7d_1}{n + 6|x_1|}\right) \cdot d_1 + \frac{(x_3 + x_2)(f(x_2) - f(x_1))}{n/2 + 3x_1} \leq \frac{\sqrt{n}}{3} \cdot \frac{3}{2} + O(1) < \frac{3\sqrt{n}}{4} \leq d_2$$

if n is large enough. Now Lemma 4 implies $f(x_1, P) < f(x_2, P) = f(x_3, P)$, hence x_1 will be removed. Then we are in the same situation as when initialising two individuals on the same branch. □

However, there is still a constant probability that the (2+1) EA finds both optima in polynomial expected time. This holds if the EA is initialised with its two search points on different branches, and if these two search points maintain similar fitness values throughout the run.

Theorem 2. *The (2+1) EA with fitness sharing with probability $\Omega(1)$ will find both optima in time $O(n \log n)$.*

Due to space restrictions, we only sketch the proof. Let x_1, x_2 be the two initial search points and $d_1 := n/2 - |x_1|$ and $d_2 := |x_2| - n/2$. With probability $\Omega(1)$, x_1 and x_2 are on opposite branches and have similar fitness: $\frac{3}{4}\sqrt{n} \leq d_1, d_2 \leq \sqrt{n}$.

Now, assume w. l. o. g. that when a new offspring is created and the population contains x_1, x_2, x_3 in order of their numbers of ones, that x_2 and x_3 are on the same branch. If $f(x_1) > f(x_2)$, Lemma 1 implies that $f(x_1, P) > f(x_2, P)$ and $f(x_2, P) < f(x_3, P)$ if $|x_3| > |x_2|$. Then x_1 is guaranteed to survive.

Now assume $f(x_1) \leq f(x_2)$. It is easy to derive from Lemma 4 and further arguments for $|x_3| - |x_1| > n/2$ that $f(x_1, P) > f(x_2, P)$ follows if $d_1 \geq (2/3) \cdot d_2$.

For a current population $P = \{x_1, x_2\}$ define a potential $g(P) := \min\{d_1, d_2\} - (2/3) \cdot \max\{d_1, d_2\}$. Intuitively, the potential indicates a distance to a population where the lower-fitness individual is at risk of dying. For $d_1 \leq d_2$ we have, using Lemma 4,

$$g(P) \geq \frac{\sqrt{n}}{24} \Leftrightarrow d_1 \geq \frac{2}{3} \cdot d_2 \Rightarrow f(x_1, P) > f(x_2, P).$$

For the initial population P_0 we have $g(P_0) \geq 3/4 \cdot \sqrt{n} - 2/3 \cdot \sqrt{n} \geq \sqrt{n}/12$. If $d_1 \leq d_2 - k$ for some $k \in \mathbb{N}$, the potential increases by k if d_1 increases by k . However, the potential only decreases by $2/3 \cdot k$ if d_2 increases by k . Moreover, increasing d_1 is easier than increasing d_2 as the former contains more “incorrect” bits (cf. Lemma 13 in [2]). This shows that, whenever the potential changes, it increases in expectation by $1/3$.

A straightforward application of the simplified drift theorem [8,9] shows that with probability $2^{-\Omega(\sqrt{n})}$ the potential never decreases below $\sqrt{n}/24$ in $2^{\Omega(\sqrt{n})}$ steps. So, with overwhelming probability x_1 survives until both optima are reached.

4 Population Size $\mu \geq 3$ Succeeds

A population of size $\mu = 2$ may fail, but we show that a $(\mu+1)$ EA with $\mu \geq 3$ always finds both optima in expected time $O(\mu n \log n)$.

The following lemma is an extension of the Main Lemma to the case where an individual $x_{\mu+1}$ is on the other branch compared to the rest of the population. In particular, a stronger condition is given such that $x_{\mu+1}$ will survive selection when $f(x_\mu) > f(x_{\mu+1})$. The proof is similar to the one for the Main Lemma.

Lemma 5. *Let $|x_\mu| < n/2$, $|x_{\mu+1}| > n/2$ and $f(x_\mu) > f(x_{\mu+1})$. Also let $h_\mu := n/2 - |x_\mu|$ and $h_{\mu+1} := |x_{\mu+1}| - n/2$. Then*

$$f(x_\mu) \cdot (\mu - 1) \cdot \frac{h_\mu}{h_\mu - h_{\mu+1}} + D_\mu \geq (\mu + 1) \cdot n/2 \Rightarrow f(x_{\mu+1}, P) \geq f(x_\mu, P).$$

The following lemma states that if there is a bounded number r of individuals in one optimum then they will have better shared fitness than the next sub-optimal individual. This implies that r such individuals survive in the $(\mu+1)$ EA; the same holds if there are more than r such individuals in the extended population as only one individual is being removed.

Lemma 6. *Let $P = \{x_1, \dots, x_s\}$ with $|x_1| \leq \dots \leq |x_s|$. Assume $|x_1| = \dots = |x_r| = 0 < |x_{r+1}|$ and $|x_s| < n$. If $r \leq 2$ or if both $|x_{r+1}| \geq n/2$ and $r \leq s/2$,*

then for all $1 \leq i \leq r$ we have $f(x_i, P) > f(x_{r+1}, P)$. In particular, if the current population of the $(\mu+1)$ EA contains at least two individuals 0^n , two such individuals always survive.

With these lemmas we are ready to prove the main result of this section.

Theorem 3. *Let $\mu \geq 3$. The $(\mu+1)$ EA with fitness sharing will find both optima of TWOMAX with probability 1 in expected time $O(\mu n \log n)$.*

Proof. By Lemma 1, in expected time $O(\mu n \log n)$ one of the two optima is found. W.l.o.g. we assume the 0^n optimum is found. In expected time $O(\mu)$ a clone of 0^n is created (i. e., $|x_2| = 0$) and by Lemma 6 x_1 and x_2 (or clones thereof) will survive for the rest of the run.

We show that then the individual with the largest number of ones, $x_{\mu+1}$ (or a clone thereof), will always survive. If $|x_\mu| = |x_{\mu+1}|$ then $x_{\mu+1}$ or a clone survive. If $n/2 \leq |x_\mu| < |x_{\mu+1}|$ then $f(x_{\mu+1}) > f(x_\mu)$ and the claim follows from Lemma 1. If $|x_\mu| < n/2$ then Lemma 3 implies $f(x_{s-1})(s-2) + D_{s-1} > sn/2$ (where $s = \mu + 1$). If $|x_{\mu+1}| \leq n/2$, by the Main Lemma this condition is equivalent to $f(x_{\mu+1}, P) > f(x_\mu, P)$. Otherwise, the same conclusion follows from Lemma 5 as $h_\mu/(h_\mu - h_{\mu+1}) > 1$. So, in all cases $x_{\mu+1}$ survives. The expected time for $x_{\mu+1}$ reaching 1^n is again $O(\mu n \log n)$ as in [4]. \square

Our analysis has revealed two very different behaviours. It is possible that the whole population climbs up one branch. But once a sufficiently large overall fitness value has been obtained – at the latest when two individuals have found an optimum – then the population expands towards lower fitness values as then the individuals with the smallest and the largest numbers of 1-bits always survive.

5 Too Large Offspring Population Sizes

Fitness sharing works for the $(\mu+1)$ EA, but for larger offspring populations it can have undesirable effects: if a cluster of individuals creates too many offspring, sharing decreases the shared fitness of all individuals in the cluster, and the cluster may go extinct. We consider this problem of overpopulation for $\mu = 2$ and $\lambda \geq \mu$ with $\lambda = O(1)$. In this setting we cannot guarantee convergence to populations with both optima any more, i. e., depending on λ we can lose one or even both optima.

Assume that all individuals are in the same optimum. With probability $\Omega(1)$, we create $\lambda - 1$ copies and one point with distance 1 to the optimum. Then, $f(x_1, P) = \dots = f(x_{\lambda+1}, P) = n/((\lambda+2) - 2/n)$ and $f(x_{\lambda+2}, P) = (n-1)/((\lambda+2) - (\lambda+1) \cdot 2/n)$. We see that $f(x_i, P) < f(x_{\lambda+2}, P)$ for all $i \in \{1, \dots, \lambda+1\}$ and $\lambda \geq 2$. Thus, selection picks $x_{\lambda+2}$ and one of the optimal points. Following the same argumentation, we lose both optima if $\lambda \geq 6$: If mutation creates $\lambda - 2$ copies and two points with distance 1 to the optimum (also with probability $\Omega(1)$), we have $f(x_1, P) = \dots = f(x_\lambda, P) = n/((\lambda+2) - 2 \cdot 2/n) < (n-1)/((\lambda+2) - \lambda \cdot 2/n) = f(x_{\lambda+1}, P) = f(x_{\lambda+2}, P)$ for $\lambda \geq 6$. In exactly the same way we show that both optima are lost with probability $\Omega(1)$ if $\lambda \geq 6$ even

if they are on different branches, i. e., we create $\lfloor \lambda/2 \rfloor$ offspring on the left branch and $\lceil \lambda/2 \rceil$ on the right branch where exactly one offspring on each branch has distance 1 to the optimum and the remaining offspring are copies.

Offspring populations can also decrease diversity in the following way.

Lemma 7. *With probability $1 - o(1)$ the $(2 + \lambda)$ EA with fitness sharing, $\lambda \geq 2$ and $\lambda = O(1)$ will, at some point of time before an optimum is reached, obtain a population with both members on the same branch.*

Proof (Proof sketch). The proof mainly uses that in a single iteration with probability $\Omega(1)$ only copies of x_1 and x_2 are created. We show that if $f(x_1) \neq f(x_2)$ and if we have a surplus of offspring on the branch with smaller fitness (also probability $\Omega(1)$), this branch goes extinct. If $f(x_1) = f(x_2)$ in iteration t we have $f(x_1) \neq f(x_2)$ in iteration $t + 1$ with probability $\Omega(1)$ and if $f(x_1) \neq f(x_2)$ in iteration t we still have $f(x_1) \neq f(x_2)$ in iteration $t + 1$ with probability $\Omega(1)$. Thus, with probability $1 - 2^{-\Omega(n)}$ there are $\Omega(n)$ iterations with $f(x_1) \neq f(x_2)$ before an optimum is reached and consequently, with probability $1 - 2^{-\Omega(n)}$, one branch will take over the whole population before an optimum is reached. \square

In order to show that the $(2 + \lambda)$ EA also reaches a population with both members in the same optimum we additionally need to show that the population will not be stuck somewhere on the branch and that individuals cannot traverse back to the other branch. We consider this for the special case of $\lambda = 2$.

Theorem 4. *With probability $1 - o(1)$ the $(2+2)$ EA with fitness sharing will, at some point of time, reach a population with both members in the same optimum. The expected time for finding both optima from there is $\Omega(n^{n/2})$.*

Proof (Proof sketch). Due to Lemma 7 both individuals are on the same branch with probability $1 - o(1)$ before an optimum is reached.

We show that a current best individual is never lost. Due to Lemma 1 $f(x_1, P) > f(x_2, P)$ holds. We apply Lemma 2 and have $f(x_3, P) \geq f(x_2, P) \Leftrightarrow D_2 \geq 2n$ where $D_2 = d_{2,1} + d_{2,3} + d_{2,4}$ since $d_{2,2} = 0$. Since all individuals are on the same branch $d_{i,j} \leq n/2$. This implies that $D_2 \leq 3n/2$ and thus, $f(x_3, P) < f(x_2, P) < f(x_1, P)$. Thus, a single best individual will always survive. Moreover, in case of 2 best individuals at least one of them will be selected for the next iteration. Since $\mu = 2$ we are guaranteed to select at least one of the best individuals if there are 3 or 4 best. Following the same argumentation, we see that a single improved offspring of a best individual will always be accepted. Thus, we will reach a population with both members in the same optimum. The claim about the expected time to find both optima follows as in Theorem 1. \square

6 Experiments

Our final contribution is a set of experiments, shown in Table 1, where we ran $(\mu + \lambda)$ EAs for $n = 100$ bits and varying values of $2 \leq \mu \leq 12$ and $1 \leq \lambda \leq 12$. We recorded the success rate as the number of runs where both optima were

Table 1. Success rates of the $(\mu+\lambda)$ EA with fitness sharing on TwOMax in 1000 runs, stopped after 100000 generations, and once both optima were found

μ	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 4$	$\lambda = 5$	$\lambda = 6$	$\lambda = 7$	$\lambda = 8$	$\lambda = 9$	$\lambda = 10$	$\lambda = 11$	$\lambda = 12$
2	0.23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	0.277	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	0.602	0.32	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	0.793	0.644	0.025	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	1.0	1.0	0.824	0.687	0.261	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	1.0	0.936	0.861	0.768	0.156	0.0	0.0	0.0	0.0	0.0	0.0
8	1.0	1.0	1.0	0.926	0.874	0.816	0.064	0.0	0.0	0.0	0.0	0.0
9	1.0	1.0	1.0	0.996	0.957	0.894	0.828	0.039	0.0	0.0	0.0	0.0
10	1.0	1.0	1.0	1.0	0.972	0.957	0.918	0.843	0.032	0.0	0.0	0.0
11	1.0	1.0	1.0	1.0	1.0	0.98	0.945	0.929	0.805	0.02	0.001	0.0
12	1.0	1.0	1.0	1.0	1.0	0.99	0.978	0.972	0.945	0.738	0.029	0.0

found within 100000 generations. The table shows a clear distinction between efficient and inefficient behaviour: for $\lambda < \lfloor \mu/2 \rfloor$ runs were always successful, whereas runs for $\lambda \geq \mu$ always failed (except for one run with $\lambda = \mu = 11$).

References

1. Cioppa, A.D., Stefano, C.D., Marcelli, A.: On the role of population size and niche radius in fitness sharing. *IEEE Trans. Evol. Comput.* 8(6), 580–592 (2004)
2. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. *Algorithmica* 64, 673–697 (2012)
3. Fischer, S., Wegener, I.: The one-dimensional Ising model: Mutation versus recombination. *Theor. Comput. Sci.* 344(2–3), 208–225 (2005)
4. Friedrich, T., Oliveto, P.S., Sudholt, D., Witt, C.: Analysis of diversity-preserving mechanisms for global exploration. *Evol. Comput.* 17(4), 455–476 (2009)
5. Goldberg, D.E.: *Genetic Algorithms for Search, Optimization, and Machine Learning*. Addison-Wesley (1989)
6. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal-function optimization. In: *Proc. of ICGA*, pp. 41–49. Lawrence Erlbaum Associates (1987)
7. Mahfoud, S.W.: Niching methods. In: Bäck, T., Fogel, D.B., Michalewicz, Z., (eds.) *Handbook of Evolutionary Computation*, pp. C6.1:1–C6.1:4. IOP Publishing and Oxford University Press (1997)
8. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica* 59(3), 369–386 (2011)
9. Oliveto, P.S., Witt, C.: Erratum: Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *ArXiv e-prints* (2012)
10. Oliveto, P.S., Zarges, C.: Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. In: *Proc. of GECCO*, pp. 837–844. ACM (2013)
11. Sudholt, D.: Crossover is provably essential for the Ising model on trees. In: *Proc. of GECCO*, pp. 1161–1167. ACM Press (2005)

Runtime Analysis of Evolutionary Algorithms on Randomly Constructed High-Density Satisfiable 3-CNF Formulas

Andrew M. Sutton¹ and Frank Neumann²

¹ Friedrich-Schiller-Universität Jena, 07743 Jena, Germany

² Optimisation and Logistics, School of Computer Science,
The University of Adelaide, Adelaide, SA, 5005, Australia

Abstract. We show that simple mutation-only evolutionary algorithms find a satisfying assignment on two similar models of random planted 3-CNF Boolean formulas in polynomial time with high probability in the high constraint density regime. We extend the analysis to random formulas conditioned on satisfiability (i.e., the so-called filtered distribution) and conclude that most high-density satisfiable formulas are easy for simple evolutionary algorithms. With this paper, we contribute the first rigorous study of randomized search heuristics from the evolutionary computation community on well-studied distributions of random satisfiability problems.

1 Introduction

Boolean satisfiability is an archetypical NP-complete problem with extensive theoretical and practical relevance. Randomized search heuristics such as evolutionary algorithms [6] and randomized local search techniques [10] are often successfully applied to quickly identify satisfiable Boolean formula. Modern high-performance heuristics can handle problems with millions of variables [13], but the relationship between problem structure and computational cost is still poorly understood from a rigorous perspective.

In the field of Boolean satisfiability, a significant amount of research has been carried out on the runtime of algorithms over randomly generated formulas. Theoretical and empirical work on uniform random satisfiability suggests that, despite the hardness of the problem of determining whether or not a Boolean formula has a satisfying assignment, a vast fraction of formulas are easy to solve on average. Understanding the behavior of evolutionary algorithms with respect to their runtime for this central problem pushes forward the theoretical understanding of these algorithms on an NP-hard problem in the context of randomly generated instances.

Extensive progress has already been made in runtime analysis of evolutionary algorithms from a worst-case perspective [1,12]. However, still very little is known about typical behavior on randomly generated instances of NP-hard problems. The only study that we are aware of is the one of Witt [16] for makespan scheduling. In this paper we study the behavior of simple evolutionary algorithms over

uniform distributions of satisfiable 3-CNF formulas. We prove that the runtime of the (1+1) EA is $O(n^2 \log n)$ with high probability on almost all satisfiable 3-CNF formulas (except for a fraction that vanishes exponentially fast), as long as their constraint density is $\Omega(n)$. Though this distribution is previously known to be easy for classical algorithms [9], to our knowledge this constitutes the first rigorous analysis of evolutionary algorithms on random satisfiability models.

1.1 3-CNF Distributions

A k -CNF formula F over a set of n Boolean variables $\{x_1, x_2, \dots, x_n\}$ is a conjunction of exactly m clauses $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$, where each clause is the disjunction of exactly k literals, $C_i = \ell_{i_1} \vee \dots \vee \ell_{i_k}$, and each literal ℓ_{i_j} is either an occurrence of a variable x or its negation \bar{x} . A k -CNF formula is *satisfiable* if and only if there is an assignment of variables to truth values so that every clause contains at least one true literal. The *constraint density* of a formula is the ratio of clauses to variables m/n . The constraint density quantifies the average number of constraints (disjunctive clauses) in which a variable occurs.

The set of all assignments to a set of n Boolean variables is isomorphic to $\{0, 1\}^n$ by interpreting each position of the string as the state of exactly one Boolean variable x_i (i.e., a 1 corresponds to $x_i = \text{true}$; a 0 corresponds to $x_i = \text{false}$). Given a 3-CNF formula F with n variables, we represent candidate solutions as length- n bitstrings and define the function $f: \{0, 1\}^n \rightarrow \mathbb{N}$ where $f(x)$ counts the clauses of F that are satisfied under the assignment corresponding to $x \in \{0, 1\}^n$. If F is satisfiable, the task of finding a satisfying assignment is reduced to the task of optimizing a pseudo-Boolean function.

Uniform distributions of random Boolean formulas are similar to the Erdős-Rényi model of random graphs. In the $\mathcal{U}_{n,m}$ model, exactly m random clauses are selected independently and uniformly with replacement¹ from all possible 3-CNF clauses over n variables. In the $\mathcal{U}_{n,p}$ model, each 3-CNF clause over n variables is chosen for inclusion independently with probability p .

Stochastic search algorithms such as evolutionary algorithms and randomized local search are generally incapable of proving a formula unsatisfiable, but are often applied as incomplete heuristics and can be treated as Monte Carlo algorithms when their runtime is fixed. Because of this, one is often interested in their performance on *satisfiable* formulas.

One way to generate random satisfiable formulas is to condition the distribution $\mathcal{U}_{n,m}$ on satisfiability. This results in the *filtered uniform* model $\mathcal{U}_{n,m}^{\text{SAT}}$. The filtered uniform model is difficult to analyze, and potentially hard to sample from (since it requires solving an NP-hard problem to check whether a formula is satisfiable). To circumvent this, *uniform planted* models attempt to “hide” a satisfiable assignment in an instance. In this model, a *planted assignment* x^* is first selected uniformly at random from $\{0, 1\}^n$, and clauses are selected uniformly from the set of all clauses that are satisfied by x^* . In the $\mathcal{P}_{n,m}$ model, exactly

¹ Generating the clause set with replacement is easier in practice, and facilitates our analysis later in the paper.

m random clauses are selected independently and uniformly with replacement from the set of clauses satisfied by x^* . Similarly, in the $\mathcal{P}_{n,p}$ model, each 3-CNF clause over n variables that is satisfied by the planted assignment is selected independently with probability p . Other conditional distributions have also been studied, for example see Krivelevich et al. [8].

1.2 Background

The (1+1) EA has been the subject of the first analyses of worst-case expected runtime for pseudo-Boolean functions. Droste et al. [5] showed that the expected runtime of the (1+1) EA is bounded above by $O(n^n)$ steps over all pseudo-Boolean functions. Moreover, they showed that linear pseudo-Boolean polynomials are optimized in $O(n \log n)$ steps in expectation by the (1+1) EA. More recently, Witt [17] has derived an $en \ln n + O(n)$ bound for the (1+1) EA optimizing linear functions, which is tight up to lower order terms.

The case of general functions over $\{0, 1\}^n$ is currently less clear. The class of pseudo-Boolean polynomial functions of degree at most $k \geq 2$ is already NP-hard since it contains maximum k -satisfiability. Some theoretical analyses have been carried out to investigate large-scale search space properties for k -satisfiability [14,15]. To our knowledge, no results connecting the k -satisfiability search space to EA runtime analysis have yet been carried out.

In this paper, we reduce the problem of finding a satisfying assignment to a 3-CNF formula to finding the maximum of a degree-3 pseudo-Boolean polynomial. Koutsoupias and Papadimitriou [7] showed that the $\mathcal{P}_{n,p}$ distribution has desirable search space properties for a greedy algorithm. Using similar techniques, we extend this analysis to the $\mathcal{P}_{n,m}$ distribution, and prove that the (1+1) EA can also exploit these properties to solve high-density formulas efficiently. For each distribution $\mathcal{P}_{n,m}$, $\mathcal{P}_{n,p}$, and $\mathcal{U}_{n,m}^{\text{SAT}}$ in the high density regime, we prove that the (1+1) EA can find a satisfying assignment in polynomial time with probability $1 - o(1)$ on every formula, except for a set of measure vanishing exponentially fast in n . We also give a corresponding lower bound that suggests our upper bounds are tight up to a factor of $O(n)$ and conjecture that our upper bounds can be improved by a linear factor.

2 Preliminaries

A sequence of events $\{\mathcal{E}_n\}$ is said to hold *with high probability* if $\lim_{n \rightarrow \infty} \Pr(\mathcal{E}_n) = 1$. We will often make use of the following theorem. A proof can be found, for example, in the text by Motwani and Raghavan [11].

Theorem 1 (Chernoff Bounds). *Let X_1, X_2, \dots, X_n be independent Poisson trials such that for $1 \leq i \leq n$, $\Pr(X_i = 1) = p_i$, where $0 < p_i < 1$. Let $X = \sum_{i=1}^n X_i$, $\mu = \mathbb{E}(X) = \sum_{i=1}^n p_i$. Then for $0 < \delta \leq 1$, $\Pr(X \geq (1+\delta)\mu) \leq e^{-\mu\delta^2/3}$ and $\Pr(X \leq (1-\delta)\mu) \leq e^{-\mu\delta^2/2}$.*

Chernoff bounds provide sharp bounds for tail probabilities in situations where we can estimate the expected number of successes from a series of independent trials. We will need the following two definitions.

Definition 1. For any arbitrary $x \in \{0, 1\}^n$, we define a pair of sets S_x and U_x that partition the set of all possible 3-CNF clauses on n variables as follows. S_x is the set of all 3-CNF clauses on n variables that are satisfied by x . Similarly, U_x is the set of all 3-CNF clauses on n variables that are not satisfied by x .

Definition 2. The hypercube graph of order n is the undirected graph $G = (V, E)$ where $V = \{0, 1\}^n$ and $\{x, y\} \in E \iff |\{i : x_i \neq y_i\}| = 1$.

Let F be a satisfiable 3-CNF formula on n variables. Denote as $x^* \in \{0, 1\}^n$ an assignment (possibly unique) that satisfies F . We define the potential function $\varphi(x) = |\{i : x_i \neq x_i^*\}|$. F induces an orientation and an edge labeling on G in the following way. Let G_{F, x^*} be the directed, edge-labeled graph such that the directed edge (x, y) appears in $E(G_{F, x^*})$ if and only if x and y are neighbors in G and $\varphi(y) < \varphi(x)$. Furthermore, (x, y) is labeled *deceptive* if x satisfies at least as many clauses in F as y .

3 Random Planted Formulas

In this section, we study the distribution of graphs G_{F, x^*} where F is a formula constructed by a random planted model and x^* is the planted assignment. We will rely on these results to apply multiplicative drift theorems that bound the runtime of the (1+1) EA for all but a vanishing fraction of high-density formula.

Definition 3. An assignment x is bad if, for any constant $\epsilon > 0$, $\varphi(x) > (1/2 + \epsilon)n$. An assignment x is good if it is not bad.

Definition 4. The directed graph G'_{F, x^*} is the subgraph of G_{F, x^*} induced by the set of all good assignments.

For a particular 3-CNF distribution, we want to derive the probability of deceptive edges appearing in the hypercube graph induced by formulas drawn from that distribution. If a region of the search space contains no deceptive edges, then the local gradient is consistent with the distance to a solution since every strictly improving Hamming neighbor of a solution in that region is also strictly closer to x^* . This is obviously a nice property to have in the search space, and we call formulas *well-structured* that have this property.

Definition 5. A planted 3-CNF formula F is said to be well-structured if there are no deceptive edges in G'_{F, x^*} where x^* is the planted assignment.

Koutsoupias and Papadimitriou [7] studied the $\mathcal{P}_{n,p}$ distribution, and the next theorem follows from their work.

Theorem 2 (Koutsoupias and Papadimitriou [7]). *Suppose F is a 3-CNF formula constructed by the $\mathcal{P}_{n,p}$ model. The probability that F is well-structured is bounded below by $1 - e^{-cpn^2 + \Theta(n)}$ for some constant $c > 0$.*

We extend this analysis to the $\mathcal{P}_{n,m}$ distribution. In this model, we first choose an assignment x^* uniformly at random, then choose exactly m clauses with replacement from the set of $(2^3 - 1)\binom{n}{3}$ clauses that satisfy x^* .

Lemma 1. *Suppose (x, y) is a directed edge in G'_{F,x^*} . Then,*

$$|S_{x^*} \cap (U_x \cap S_y)| = \binom{n-1}{2}, \text{ and, } |S_{x^*} \cap (S_x \cap U_y)| \leq \gamma(n) \binom{n-1}{2}$$

where $\gamma(n) = (1 + o(1))(3/4 + \epsilon - \epsilon^2)$ for any constant $\epsilon > 0$.

Proof. Without loss of generality, suppose $x^* = (1, 1, \dots, 1)$. In this case, S_{x^*} is the set of all clauses with at least one positive literal. Since x and y are Hamming neighbors, they differ by exactly one bit i which is set to zero in x and set to 1 in y . Thus $S_{x^*} \cap (U_x \cap S_y)$ contains clauses where (1) x_i appears as a positive literal, and (2) the polarity of the remaining two literals in the clause are uniquely determined by their state in x . There are $\binom{n-1}{2}$ ways to choose these remaining two literals. Similarly, $S_{x^*} \cap (S_x \cap U_y)$ contains clauses in which the literal \bar{x}_i appears and the polarity of the remaining two literals again are uniquely determined. However, we cannot choose all $\binom{n-1}{2}$ such literals, because some of these correspond to clauses where all three literals are negative (and hence do not belong to S_{x^*}). These literals correspond to the elements in x that are set to 1 since all such literals must be negative if they appear in any clause not satisfied by y . There are $n - \varphi(x)$ such elements. By subtracting out the $\binom{n-\varphi(x)}{2}$ ways to choose two negative literals, we obtain

$$|S_{x^*} \cap (S_x \cap U_y)| = \binom{n-1}{2} - \binom{n-\varphi(x)}{2} \leq \binom{n-1}{2} - \binom{n(1/2 - \epsilon)}{2}.$$

The final inequality holds since x is good, so $\varphi(x) \leq n(1/2 + \epsilon)$. Setting

$$\gamma(n) = 1 - \binom{n(1/2 - \epsilon)}{2} / \binom{n-1}{2}$$

completes the proof since $\lim_{n \rightarrow \infty} \gamma(n) = 3/4 + \epsilon - \epsilon^2$. □

Theorem 3. *Suppose F is a 3-CNF formula constructed by the $\mathcal{P}_{n,m}$ model. The probability that F is well-structured is bounded below by $1 - e^{-cm/n + \Theta(n)}$ for some constant $c > 0$.*

Proof. Let (x, y) be an arbitrary edge in G'_{F,x^*} . We define the following random variables that count clauses in F .

$$\begin{aligned} Z_1 &= |\{\text{clauses } C \text{ in } F : C \in (S_{x^*} \cap (U_x \cap S_y))\}|, \\ Z_2 &= |\{\text{clauses } C \text{ in } F : C \in (S_{x^*} \cap (S_x \cap U_y))\}|. \end{aligned}$$

Since the m clauses are chosen independently with replacement, the probability of choosing a clause from $(S_{x^*} \cap (U_x \cap S_y))$ is, by Lemma 1, $\binom{n-1}{2} / \binom{n}{3} = 3/(7n)$. Similarly, the probability of choosing a clause from $(S_{x^*} \cap (S_x \cap U_y))$ is at most $3\gamma(n)/(7n)$. Hence Z_1 and Z_2 are binomially distributed independent random variables, both with m trials, and their expected values are $E(Z_1) = 3m/(7n)$ and $E(Z_2) \leq \gamma(n)3m/(7n)$.

The event that (x, y) is labeled deceptive under F is equivalent to the event $Z_1 \leq Z_2$, and thus the probability that (x, y) is labeled deceptive is $\Pr(Z_1 \leq Z_2) \leq \Pr(Z_1 \leq t) + \Pr(Z_2 \geq t)$ for any $t > 0$. Appealing to Theorem 1, this is at most

$$\begin{aligned} & \exp\left(-\frac{(t - E(Z_1))^2}{2E(Z_1)}\right) + \exp\left(-\frac{(t - E(Z_2))^2}{3E(Z_2)}\right) \\ & \leq \exp\left(-\frac{(t - E(Z_1))^2}{3E(Z_1)}\right) + \exp\left(-\frac{(t - E(Z_2))^2}{3E(Z_2)}\right). \end{aligned}$$

Setting $t = \sqrt{E(Z_1)E(Z_2)}$, the probability is at most

$$2 \exp\left(-\frac{(\sqrt{E(Z_1)} - \sqrt{E(Z_2)})^2}{3}\right) \leq 2 \exp\left(-\frac{m(1 - \sqrt{\gamma(n)})^2}{7n}\right) < 2e^{-cm/n},$$

by substituting the value bounds on the expectations of Z_1 and Z_2 from above. Here $0 < c < (1 - \sqrt{\gamma(n)})^2 / 7$ is a positive constant following from the asymptotic bound on $\gamma(n)$.

Finally, by applying the union bound, the probability that any edge in G'_{F,x^*} is deceptive is at most $|E|2e^{-cm/n}$. The claim then follows from the fact that the number of edges in G'_{F,x^*} is at most $n2^{n-1}$. □

The uniform filtered 3-CNF distribution $\mathcal{U}_{n,m}^{\text{SAT}}$ is the conditional distribution generated by conditioning $\mathcal{U}_{n,m}$ on satisfiability. For dense enough formulas, the uniform filtered distribution is statistically close to the planted distribution.

Theorem 4 (Ben-Sasson et al. [2]). *The 3-CNF distributions $\mathcal{P}_{n,m}$ and $\mathcal{U}_{n,m}^{\text{SAT}}$ coincide in the regime $m/n = \Omega(\log n)$ in the following sense.*

There exists a constant $c > 0$ such that when $m \geq cn \ln n$, then with high probability, a formula constructed by the $\mathcal{P}_{n,m}$ or the $\mathcal{U}_{n,m}^{\text{SAT}}$ model has exactly one satisfying assignment. Moreover, if F is an arbitrary formula with m clauses and n variables, such that F has a unique satisfying assignment, the probability of constructing F from $\mathcal{P}_{n,m}$ is equal to the probability of constructing F from $\mathcal{U}_{n,m}^{\text{SAT}}$.

Hence for $m/n = \Omega(\log n)$, except for a set of measure that tends to zero, formulas constructed by the planted model or the filtered model have the same probability. It follows that the claim of Theorem 3 also applies to $\mathcal{U}_{n,m}^{\text{SAT}}$ in the high-density regime ($m/n \geq cn$ for a constant $c > 0$ sufficiently large).

4 Runtime Analysis

The runtime analysis of randomized search heuristics on randomly constructed instances involves two sources of randomness. We must deal with random inputs, in this case, the random formula, and also with the random decisions of the algorithm at the same time. To handle this, we assume the formula has the well-structured property and derive tail bounds on the runtime conditioned on that property. We then use the results of the previous section to bound the probability that the formula is well-structured in a given density regime.

We analyze the runtime of the standard (1+1) EA (Algorithm 1) searching for a satisfying assignment to a formula F by optimizing the corresponding pseudo-Boolean function f that counts the satisfied clauses in F .

Algorithm 1. The (1+1) EA

```

choose  $x \in \{0, 1\}^n$  uniformly at random;
repeat forever
   $y \leftarrow x$ ;
  flip each bit of  $y$  independently with prob.  $1/n$ ;
  if  $f(y) \geq f(x)$  then  $x \leftarrow y$ 

```

Following the typical approach to runtime analysis, we view each run of the (1+1) EA as an infinite stochastic process $(x^{(1)}, x^{(2)}, \dots, x^{(t)}, \dots)$, where $x^{(t)} \in \{0, 1\}^n$ denotes the assignment generated in iteration t of the algorithm. The runtime T of an algorithm is the random variable $T = \inf\{t \in \mathbb{N} : x^{(t)} \text{ satisfies } F\}$. The main result of this section is stated in the following theorem.

Theorem 5. *Suppose F is a well-structured formula. Then with probability $1 - o(1)$, the time until the (1+1) EA finds a satisfying assignment for F is bounded by $O(n^2 \log n)$.*

To prove Theorem 5, we will rely on the favorable search space properties of well-structured formulas. In particular, we will show that, as long as the (1+1) EA remains in the good region, its drift towards the planted assignment can be bounded below by a positive term.

Lemma 2. *Suppose F is a well-structured formula and that $\varphi(x^{(t)}) \leq (1/2 + \epsilon/2)n$. Then the probability that $x^{(t+1)}$ is a bad assignment is at most $e^{-\Omega(n \log n)}$.*

Moreover, if $\varphi(x^{(1)}) \leq (1/2 + \epsilon/2)n$, then with probability $1 - o(1)$, after $t \leq p(n)$ iterations, where p is a polynomial in n , the (1+1) EA never generates a bad assignment.

Proof. In each step, the probability that at least k bits are changed is at most

$$\binom{n}{k} \left(\frac{1}{n}\right)^k \leq \frac{1}{k!} \leq \left(\frac{e}{k}\right)^k = e^{-\Omega(k \log k)}.$$

The assignment $x^{(t)}$ is at Hamming distance at least $n\epsilon/2$ from any bad assignment. Thus, for $x^{(t+1)}$ to be bad, mutation must change at least $k = n\epsilon/2$ bits.

The second part of the claim follows from the fact that the probability of no bad assignment generated in $p(n)$ iterations is at least

$$\left(1 - e^{-\Omega(n \log n)}\right)^{p(n)} \geq 1 - p(n) \cdot e^{-\Omega(n \log n)} = 1 - o(1),$$

where we have applied Bernoulli’s inequality. We remark here that even after any polynomial number of steps, the probability that the (1+1) EA never generates a bad assignment is going to one exponentially fast. \square

Lemma 3. *We consider the execution of the (1+1) EA on a well-structured formula F . Define the sequence of random variables $\{X_t : t > 0\}$ as $X_t = \varphi(x^{(t)})$. We bound the drift of the stochastic process described by this sequence from below. Suppose that $\varphi(x^{(t)}) \leq (1/2 + \epsilon/2)n$, then $E(X_t - X_{t+1} \mid X_t) \geq cX_t/n^2$ where c is a positive constant.*

Proof. Without loss of generality, let $x^* = (1, 1, \dots, 1)$. We consider the contribution to the drift from different events. Let y be the intermediate offspring produced by mutating $x^{(t)}$. Note that by the dynamics of the (1+1) EA, $x^{(t+1)} = y$ if and only if $f(y) \geq f(x^{(t)})$.

Let A denote the event that $\varphi(y) > (1/2 + \epsilon)n$. In this event, the drift can be negative if $f(y)$ is no worse than $f(x^{(t)})$. By the law of total expectation, the drift can be written as

$$E(X_t - X_{t+1} \mid X_t \cap \neg A)(1 - \Pr(A)) + E(X_t - X_{t+1} \mid X_t \cap A)\Pr(A).$$

Moreover, we have assumed that $\varphi(x^{(t)}) \leq (1/2 + \epsilon/2)n$ so $\Pr(A)$ can be bounded by Lemma 2, and we thus have

$$E(X_t - X_{t+1} \mid X_t) \geq (1 - o(1))E(X_t - X_{t+1} \mid X_t \cap \neg A) - ne^{-\Omega(n \log n)}. \quad (1)$$

For the remaining cases of the proof, we assume that the event $\neg A$ has occurred. This is equivalent to the assumption that y lies in the good region. Let B be the event that at least one of the $\varphi(x^{(t)})$ zero-bits flip. Since we assume that both $x^{(t)}$ and y are in the good region, we now argue that if the event $\neg B$ occurs, then either $y = x^{(t)}$, or $f(y) < f(x^{(t)})$. Under this event, if none of the $n - \varphi(x^{(t)})$ one-bits flip to zero, then obviously y and $x^{(t)}$ are equivalent. On the other hand, if some one-bits flip, by transitivity of non-deceptive edges in G_{F,x^*} , y must satisfy strictly fewer clauses than $x^{(t)}$. In either case, after selection $x^{(t)} = x^{(t+1)}$. By the law of total probability we have

$$E(X_t - X_{t+1} \mid X_t \cap \neg A) = \Pr(B)E(X_t - X_{t+1} \mid X_t \cap \neg A \cap B)$$

since the drift is zero under the event $\neg B$. Since each one-bit flips with probability $1/n$, by linearity of expectation,

$$E(X_t - X_{t+1} \mid X_t \cap \neg A \cap B) \geq \left(1 - \frac{n - \varphi(x^{(t)})}{n}\right) = \frac{X_t}{n}.$$

Finally, since $\varphi(x^{(t)}) \geq 1$ (otherwise, a satisfying assignment has been found) $\Pr(B) \geq 1/n$. The claim is then proved by applying Equation (1) and choosing a sufficiently small constant c . \square

Proof of Theorem 5. By Theorem 1, with high probability $\varphi(x^{(1)}) \leq (1/2 + \epsilon/2)n$. Lemma 3 ensures that the drift of the stochastic process defined by the potential function is multiplicative by a factor bounded by $\Omega(1/n^2)$. Applying the well-known Multiplicative Drift Theorem [4], as long as the (1+1) EA never jumps out of the good region, it has reduced the potential to zero in $O(n^2 \log n)$ steps. Furthermore, this bound holds with probability $1 - o(1)$ over the run [3].

Appealing to Lemma 2, after $O(n^2 \log n)$ iterations, the (1+1) EA generates any bad assignment only with probability $o(1)$ (and this term is even vanishing exponentially fast), hence the claim is proved. \square

Corollary 1. *There exist positive constants c_1 and c_2 such that if F is a 3-CNF formula constructed from (1) the $\mathcal{P}_{n,p}$ model with $p \geq c_1/n$, or (2) the $\mathcal{P}_{n,m}$ model (and, due to Theorem 4, the $\mathcal{U}_{n,m}^{\text{SAT}}$ model) with $m \geq c_2 n^2$, then the (1+1) EA has found a satisfying assignment in $O(n^2 \log n)$ steps with probability $1 - o(1)$.*

As we have already seen in the claim of Theorem 4, high density satisfiable random formulas are likely to have exactly one satisfying assignment. In such a case, it is straightforward to derive a lower bound on the expected runtime of the (1+1) EA. In particular, with probability $1/2$, the randomly generated initial solution differs from the unique assignment in at least half the bits. Each such bit must flip at least once during the run until the satisfying assignment is found, and the expected number of steps before this event occurs is bounded below by $\Omega(n \log n)$. This bound is derived in Lemma 10 of the paper by Droste et al. [5] and immediately proves the following theorem.

Theorem 6. *If F is a random planted 3-CNF formula constructed as in Corollary 1, then with high probability F has exactly one satisfying assignment. In this case, the expected runtime of the (1+1) EA on F is bounded below by $\Omega(n \log n)$.*

5 Conclusion

In this paper, we have proved that all but a vanishing fraction of high-density random planted 3-CNF formulas can be solved efficiently by the (1+1) EA. We have shown that in the high-density regime, constraints impose favorable structure on the search space explored by such algorithms so that they run in polynomial time. In particular, we proved that the (1+1) EA finds a satisfying assignment in $O(n^2 \log n)$ iterations with probability $1 - o(1)$ on the $\mathcal{P}_{n,p}$ model when $p \geq c_1/n$ and on the $\mathcal{P}_{n,m}$ model when $m/n \geq c_2 n$ for sufficiently large positive constants c_1 and c_2 . Since, at high densities, the $\mathcal{P}_{n,m}$ distribution is statistically close to the uniform filtered $\mathcal{U}_{n,m}^{\text{SAT}}$ distribution, our results carry over to this case as well.

Additionally, we have presented a rigorous argument that the (1+1) EA takes at least $\Omega(n \log n)$ steps in expectation to solve all but a $o(1)$ fraction of random satisfiable 3-CNF formulas at high densities. We conjecture that the upper bound can be tightened to match this lower bound, and leave this as an open problem.

Acknowledgments. The research leading to these results has received funding from the Australian Research Council (ARC) under grant agreement DP140103400 and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 618091 (SAGE).

References

1. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics: Foundations and Recent Developments. World Scientific Publishing Company (2011)
2. Ben-Sasson, E., Bilu, Y., Gutfreund, D.: Finding a randomly planted assignment in a random 3-CNF (2002) (manuscript)
3. Doerr, B., Goldberg, L.A.: Drift analysis with tail bounds. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 174–183. Springer, Heidelberg (2010)
4. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. *Algorithmica* 64(4), 673–697 (2012)
5. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276(1-2), 51–81 (2002)
6. Gottlieb, J., Marchiori, E., Rossi, C.: Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation* 10(1), 35–50 (2002)
7. Koutsoupias, E., Papadimitriou, C.H.: On the greedy algorithm for satisfiability. *Information Processing Letters* 43(1), 53–55 (1992)
8. Krivelevich, M., Sudakov, B., Vilenchik, D.: On the random satisfiability process. *Combinatorics, Probability and Computing* 18, 775–801 (2009)
9. Krivelevich, M., Vilenchik, D.: Solving random satisfiable 3CNF formulas in expected polynomial time. In: SODA, pp. 454–463 (2006)
10. Kroc, L., Sabharwal, A., Selman, B.: An empirical study of optimal noise and runtime distributions in local search. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 346–351. Springer, Heidelberg (2010)
11. Motwani, R., Raghavan, P.: Randomized algorithms. Cambridge University Press, New York (1995)
12. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity. Springer (2010)
13. Seitz, S., Orponen, P.: An efficient local search method for random 3-satisfiability. *Electronic Notes in Discrete Mathematics* 16, 71–79 (2003)
14. Sutton, A.M., Howe, A.E., Whitley, L.D.: A theoretical analysis of the k -satisfiability search space. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2009. LNCS, vol. 5752, pp. 46–60. Springer, Heidelberg (2009)
15. Sutton, A.M., Whitley, L.D., Howe, A.E.: A polynomial time computation of the exact correlation structure of k -satisfiability landscapes. In: GECCO (2009)
16. Witt, C.: Worst-case and average-case approximations by simple randomized search heuristics. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 44–56. Springer, Heidelberg (2005)
17. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability and Computing* 22(2), 294–318 (2013)

Author Index

- Aguirre, Arturo Hernández 352
Aguirre, Hernán 487, 682
Akimoto, Youhei 252
Alexander, Brad 384
Allmendinger, Richard 498, 741
Alyahya, Khulood 862
Amelio, Alessia 222
Arabas, Jaroslaw 761, 872
Arnold, Dirk V. 882
Atamna, Asma 60
Auger, Anne 60
Azad, R. Muhammad Atif 444
- Bäck, Thomas 11
Badkobeh, Golnaz 892
Baiocchi, Marco 161
Banzhaf, Wolfgang 424
Bartoli, Alberto 394
Bartz-Beielstein, Thomas 373
Batina, Lejla 812, 822
Baudiš, Petr 40
Bertasius, Gediminas 211
Bezerra, Leonardo C.T. 508
Bhalla, Navneet 751
Biedrzycki, Rafał 761, 872
Bosman, Peter A.N. 342
Bringmann, Karl 518
Brockhoff, Dimo 548
Buzdalov, Maxim 528
- Chotard, Alexandre 902
Christensen, Anders Lyhne 233
Chuang, Chung-Yao 312
Clachar, Sophie 771
Clegg, Kester Dean 692
Coello Coello, Carlos A. 652, 682
Corus, Dogan 912
Costa, Lino 538
Cotta, Carlos 50, 322, 731
Cox, Chris R. 404
Cumar, Simone 394
- Dang, Duc-Cuong 912
De Lorenzo, Andrea 394
Denysiuk, Roman 538
- Derbel, Bilel 548, 641
Desell, Travis 771
de Waard, Maarten 589
Domínguez, Ignacio Segovia 352
Dorigo, Marco 181, 751
Dorscheid, Marita 80
Drugan, Mădălina M. 559
Dvorak, Vaclav 414
- Eiben, Agoston Endre 24, 110
Emmerich, Michael T.M. 11, 672
Eremeev, Anton V. 912
Espírito Santo, Isabel 538
- Farid, Suzanne S. 498, 741
Fernandes, Carlos M. 50
Fernández-de-Vega, Francisco 702
Floreano, Dario 272
Fonseca, Carlos M. 672
Friedrich, Tobias 518, 922
Fucik, Otto 802
Fukunaga, Alex 201
- García-Valdez, Mario 702
Gerontas, Spyridon 741
Glasmachers, Tobias 569, 579
Goldingay, Harry 171
Gomes, Jorge 233
Guerreiro, Andreia P. 672
- Ha, Myoung Hoon 151
Haasdijk, Evert 110
Hamann, Heiko 181
Hansen, Nikolaus 60, 70
Harding, Simon L. 721
Hart, Emma 282
Héritier, Aurélie 262
Higgins, James 771
Holeña, Martin 902
Hrbacek, Radek 414
Hu, Bin 792
Hu, Ting 424
- Inja, Maarten 589
Ipparthy, Dhananjay 751

- Ishibuchi, Hisao 600
 Izzo, Dario 110, 262, 662, 711

 Jakobovic, Domagoj 812, 822
 Jasik, Agata 761
 Jin, Yaochu 302
 Joshi, Ayush 243

 Kendall, Graham 842
 Khaluf, Yara 181
 Klemp, Eric 751
 Klitzke, Patrick 518
 Kloimüller, Christian 792
 Koch, Patrick 292
 Konen, Wolfgang 292
 Kooijman, Chiel 589
 Kovacs, Tim 191
 Krawiec, Krzysztof 434, 611

 Labroquère, Jérémie 262
 Lane, Fergal 444
 Laredo, Juan L.J. 50
 Lehre, Per Kristian 892, 912
 Lenarčič, Jadran 1
 Lewis, Peter R. 171
 Li, Rui 11
 Liefoghe, Arnaud 487, 548, 621, 641
 Liskowski, Paweł 611
 López-Ibañez, Manuel 508, 621
 Loshchilov, Ilya 70
 Lykkebø, Odd Rune 721

 Maesani, Andrea 272
 Mambrini, Andrea 711
 Marceau-Caron, Gaetan 631
 Marchiori, Elena 822
 Mariano, Pedro 233
 Marquet, Gauvain 641
 Märtens, Marcus 662
 Massey, Kieran 692
 Massey, Mark K. 721
 Masuda, Hiroyuki 600
 Mauser, Ingo 80
 McCall, John 332
 McKay, Robert Ian 151
 Medvet, Eric 394
 Menchaca-Mendez, Adriana 652
 Merelo, Juan Julian 50
 Merelo-Guérvos, Juan Julián 702
 Milani, Alfredo 161

 Miller, Julian Francis 476, 692, 721
 Mohid, Maktuba 721
 Montero, Elizabeth 90
 Moore, Jason H. 211, 424

 Nagata, Yuichi 782
 Nakata, Masaya 191
 Nallaperuma, Samadhi 100
 Naujoks, Boris 579
 Neumann, Frank 100, 922, 942
 Nogueras, Rafael 50, 322, 731
 Nojima, Yusuke 600
 Nowak, Krzysztof 662

 Oliveto, Pietro S. 932
 Ono, Isao 782

 Papazek, Petrina 792
 Pawlak, Tomasz P. 454
 Petrlik, Jiri 802
 Petty, Michael C. 721
 Petty, Mike 692
 Picek, Stjepan 812, 822
 Pizzuti, Clara 222
 Pošík, Petr 40
 Preuss, Mike 141

 Qian, Chao 302

 Raidl, Günther R. 792, 832
 Regnier-Coudert, Olivier 332
 Riccardi, Annalisa 262
 Riff, María-Cristina 90
 Roijers, Diederik M. 589
 Rosa, Agostinho C. 50
 Rothlauf, Franz 465
 Rowe, Jonathan E. 243, 862
 Rudolph, Günter 579
 Ryan, Conor 444

 Sadowski, Krzysztof L. 342
 Santucci, Valentino 161
 Schauer, Christian 832
 Schmeck, Hartmut 80
 Schoenauer, Marc 70, 631
 Sebag, Michèle 70, 852
 Sekanina, Lukas 802
 Shalyto, Anatoly 528
 Shirakawa, Shinichi 252
 Sim, Kevin 282

- Simões, Luís F. 110
Smith, Jim 120
Smith, Stephen F. 312
Solar-Lezama, Armando 434
Sosa Hernández, Víctor A. 682
Stich, Sebastian Urban 130
Stork, Jörg 373
Stütze, Thomas 508
Sudholt, Dirk 892, 932
Sutton, Andrew M. 942
Szymański, Michał 761
- Takadama, Keiki 191
Talbi, El-Ghazali 641
Tanabe, Ryoji 201
Tanaka, Kiyoshi 487, 682
Tanigaki, Yuki 600
Thierens, Dirk 342
Thorhauer, Ann 465
Titchener-Hooker, Nigel J. 741
Trautmann, Heike 141
Trujillo, Leonardo 702
Tufte, Gunnar 721
Turner, Andrew James 476
- Urbanowicz, Ryan J. 211
Ursem, Rasmus K. 362
- Valdez, S. Ivvan 352
Valentini, Gabriele 181
Verel, Sébastien 487, 548, 621
- Wagner, Markus 100
Wang, Ji 11
Wasylczyk, Piotr 761
Watson, Richard A. 404
Wessing, Simon 141
While, Lyndon 842
Whiteson, Shimon 589
Wild, Brandon 771
Wnuk, Paweł 761
Wójcik-Jedlińska, Anna 761
- Yang, Kaifeng 11
Yevseyeva, Iryna 672
Yu, Yang 302
Yun, Hansang 151
- Zacher, Brad 384
Zaefferer, Martin 373
Zapotecas Martínez, Saúl 682
Zarges, Christine 243, 932
Zhang, Guohua 852
Zhou, Zhi-Hua 302